BAYESIAN METHODS FOR AUTONOMOUS LEARNING SYSTEMS

by

Jo-Anne S. Y. Ting

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

May 2009

Acknowledgements

This dissertation could not have been completed without the support and help of many. I have been very fortunate to have Stefan Schaal as an advisor, collaborator and sounding board for the many frustrations faced over the years. He has provided not only guidance and inspiration, but a nurturing and supportive environment for growth that anyone could hope for.

I must thank Gaurav Sukhatme, Nicolas Schweighofer, Sethu Vijayakumar, Laurent Itti, Fei Sha and Leana Golubchik for their invaluable research (and career) advice and feedback. I would also like to thank my fellow collaborators Kenji Yamamoto, Toshinori Yoshioka, Donna Hoffman, Shinji Kakei, Lauren Sergio, Peter Strick, Mitsuo Kawato and John Kalaska for their help with the neural-EMG data analyses. I am indebted to Sherri Fagan and Laura Lopez for all the work they have done to make the Phd-related paperwork process less painful.

All the folks at CLMC—past and present—have been a warm and welcoming family, helping me acclimatize to the sunny, concrete sprawl of Los Angeles. The "other" robotic labs on the fourth floor, RESL and Interaction Lab, have also been great contributors to providing a socially stimulating environment at USC. A special thank you goes out to

Aaron D'Souza for being such a wonderful mentor and friend and to Mrinal Kalakrishnan, who has been a great help in the past year.

I could not have survived the last five years without those who came along for fun outdoor (beach, mountain & road) excursions and sing-and-drink distractions from research. I would like especially to thank the Baker family—Sue, Teresa and Benji—and the Brasca family—Angiola and Carlo—for making me feel so welcomed in Los Angeles and Milan, respectively, and for being my "family away from family".

I am deeply grateful for the encouragement of my parents (Sharon and Joseph), my siblings (Michele, Mike and Adrian) and friends who, even from afar, have offered a ready ear, insightful words and unlimited moral support whenever it was needed. I would like to thank Leslie Cheng, Yiffie Zhu and Madelaine Saginur for being truly fantastic friends over the years. My final thank you goes to Claudio Brasca for his love and patience.

Table of Contents

List Of Tables

List Of Figures

Abstract

We propose a set of Bayesian methods that help us toward the goal of autonomous learning systems. Systems that can react autonomously, with minimal human supervision, have the potential for significant impact, especially in applications with considerable uncertainty in the environment. In current algorithms, parameter values are set using heuristic techniques, statistical cross-validation or other search procedures to find the "right" values. We rely on Bayesian inference as a principled way to eliminate open parameters, resulting in a black-box-like approach.

We are interested in scenarios where the input data is high-dimensional (and many input dimensions may be redundant or irrelevant) and where real-time, incremental learning may be needed. Such data sets are common in the domain of robotics and brain-machine interfaces, which are the main areas of evaluation in this dissertation. We start by examining the problem of regression since classification can be performed by interpreting regression outputs in a binary way.

This dissertation first introduces a set of autonomous Bayesian methods that learn from data with the following properties: a high number of input dimensions, noise in input data, and outliers. All these methods can be leveraged together to develop a local Bayesian kernel shaping framework for nonlinear regression. The Bayesian kernel

shaping algorithm we propose is the first step towards realizing real-time autonomous learning systems. Even though the version described in this thesis is in batch form, it is computationally efficient and can be used in not only local methods, but also global nonlinear methods such as Gaussian processes for non-stationary function approximation.

Chapter 1

Introduction

1.1 Autonomous Learning

A long-standing dream of machine learning is to create autonomous learning systems that can operate, with minimal human supervision, in home, research and industrial applications. An autonomous learning system learns from experience, not requiring any application-specific tuning of parameters by the human user, while having a "black-box"-like ability to work across different hardware platforms and environments. Manual tuning parameters, such as those used in gradient descent or structure selection, need to be minimized and, ideally, avoided.

Most of the current learning algorithms require some amount of application-specific tailoring. One of the primary challenges in machine learning is to develop algorithms that are applicable across a broad range of applications with a minimal amount of modification. While a universal black box may not be possible for all systems, significant progress can be made in some domains.

In this dissertation, we address learning problems in sensor-rich and data-rich environments, such as those provided by autonomous vehicles, surveillance networks, biological systems and robotics. In these scenarios, the input data is high-dimensional and is used to make predictions, often in real-time.

1.1.1 What is Autonomous Learning?

Our goal is to devise algorithms that can deal autonomously and efficiently with high dimensional data which is typically contaminated by noise, redundancy and irrelevant dimensions. The algorithms must learn nonlinear functions, potentially in an incremental and real-time fashion (where samples are available sequentially, one at a time, instead of all together at the outset), for robust classification and regression. In this dissertation, an autonomous learning algorithm is loosely defined and considered to be an algorithm that can perform on different systems, environments and data sets, with minimal tuning of parameters, interference, and involvement by the user.

1.1.2 Autonomous Learning and Model Complexity

Given our goal of autonomous learning systems, the challenge of the learning process is to remove any tuning of model parameters needed by the user and to include it as part of the inference procedure in order to find a model that balances data fitting and model complexity, i.e., Occam's razor. Existing approaches that attempt to find this "right" amount of model complexity include cross-validation (Stone 1974, Stone & Brooks 1990), early stopping (Finnof, Hergert & Zimmerman 1993), a penalized cost function with some regularization criterion—such as minimum description length (Rissanen 1978) or the Akaike

Information Criterion (Akaike 1974), maximum-margin approaches (Taskar, Chatalbashev, Koller & Guestrin 2005), and maximizing Bayesian evidence (Mackay 1992), to name a few. It is this last approach of maximal Bayesian evidence that we choose to adopt for the following chapters.

1.1.3 Why Bayesian?

A statistical Bayesian framework offers many advantages. These include automatic complexity regularization, tractable approximate inference (when combined with approximation methods to reduce computational complexity)—which is especially important in data-rich, real-time domains, confidence measures of performance, the incorporation of domain (or prior) knowledge, and integration in other probabilistic systems.

However, Bayesian approaches require that prior distributions be chosen, and this is typically done for analytical convenience rather than real knowledge of the problem. Additionally, computationally intractable integrals arise that are hard to solve analytically (Jordan, Ghahramani, Jaakkola & Saul 1999).

1.2 Approximate Inference Methods

Various approximation methods have been proposed to cope with the intractable integrals that result from Bayesian inference. Some of these include variational approximations, e.g., (Jordan et al. 1999, Ghahramani & Beal 2000, Beal 2003), Expectation-Propagation (EP) (Minka 2001), Laplace's method (MacKay 2003, Bishop 2006) and Markov Chain Monte Carlo (MCMC) sampling methods, e.g., (Gelfand 1996).

Variational methods, EP and Laplace's method are deterministic techniques which approximate the marginal posterior distribution with a Gaussian. In contrast, Monte Carlo sampling methods are nondeterministic and rely on the law of large numbers to evaluate the integral. Some of these methods, such as EP and variational EM, are iterative. All attempt to choose the best approximation to the probability density function from within a tractable class of distributions (e.g., Gaussian, exponential, concave or convex, factorized, etc.).

Before we delve into the details of various approximate inference methods, let us introduce, in the next section, a few quantities used in Bayesian learning.

1.2.1 Bayesian Learning: Quantities of Interest

To perform Bayesian inference, let us first assume a data set D has been observed and that we want to estimate the underlying model of how the data was generated. We denote the model parameters as θ and M as the model.

In a Bayesian framework, probability distributions are placed over parameters in order to describe beliefs and uncertainties about parameter values. The prior distribution $p(\theta)$ describes the belief about the true values of θ. We can account for the information in the observed data D and use it to update the belief in θ to produce a posterior distribution $p(\theta|D)$.

The three quantities of interest in Bayesian learning consist of the following:

- Evidence for model M:

$$p(D|M) = \int p(D, \theta|M) d\theta = \int p(D|\theta) p(\theta|M) d\theta \tag{1.1}$$

Given a maximal Bayesian evidence framework, our goal is to maximize the maximize to trade off data fitting with model complexity. We can do this by maximizing the marginal likelihood, which can be arrived by integrating out model parameters from the evidence.

- Posterior distribution of parameters:

  We can use Bayes' rule to compute the posterior distribution as:

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{1.2}$$

  where $p(D|\theta)$ is the likelihood and $p(D)$ is the marginal likelihood.

- Predictive distribution:

  Given a new data sample $\mathbf{x}$, the predictive distribution is:

$$p(x|D) = \int p(x, \theta|D)d\theta = \int p(x|\theta)p(\theta|D)d\theta \tag{1.3}$$

  Notice that the posterior distribution $p(\theta|D)$ is needed to calculate the predictive distribution.

A common problem is that posterior $p(\theta|D)$ cannot be evaluated in closed form because the marginal likelihood $p(D)$ consists of an integral that is analytically intractable. Indeed, the integrals in Eq. (1.1) and (1.3) may also be hard to solve analytically. The next section describes some of the methods to approximate analytically intractable integrals.

1.2.2 Variational Approximations

Variational methods have been applied in the fields of physics, statistics and control theory in the form of calculus of variations, linear and non-linear moment problems and dynamic programming. In the recent past, variational methods have been developed for approximate inference and estimation in the probabilistic models commonly found in machine learning. Variational methods convert a complex problem into a simpler one by decoupling the degrees of freedom in the original problem. The decoupling is achieved by introducing additional parameters, known as variational parameters, that must be optimized for the problem.

In this thesis, we leverage variational approximations in order to get a tractable lower bound to the marginal log likelihood. In particular, we use two approximations: a factorial variational approximation and a variational approach on concave or convex functions.

Variational Mean Field Theory: Variational mean field theory approximates a complex distribution by fully factorizing it into individual component distributions. Mean field theory has been used by Jaakkola & Jordan (2000), Jordan et al. (1999) and Ghahramani & Beal (2000) (the latter, under the name "variational Bayes"), to name a few, for finding the lower bound to the marginal likelihood and for providing analytical approximations to parameter posterior distributions.

The factorial variational approximation can be incorporated to the Expectation-Maximization (EM) algorithm (Dempster, Laird & Rubin 1977) to optimize iteratively a lower bound to the marginal likelihood. The iterations of variational Bayesian EM consist of a variational E-step where the hidden variables are inferred using an ensemble of

models (according to their posterior probability) and a variational M-step where a posterior distribution over model parameters is inferred. Instead of adopting a fully factorized assumption of the posterior distribution, it is possible to use a more structured mean field approach, where the variational distribution factors across partitioned, disjoint sets of variables (Beal 2003).

Convex Duality: Variational transformations can also be achieved by convex duality (Rockafellar 1972) principle to obtain lower (or upper) bounds on a function. The principle of convex duality states that a concave function $f(x)$ can be represented via a conjugate or dual function as follows:

$$f(x) = \min_{\lambda} \left\{ \lambda^T f(x) - f^*(\lambda) \right\} \tag{1.4}$$

where the conjugate function $f^*(\lambda)$ can be obtained from the dual expression:

$$f^*(x) = \min_{x} \left\{ \lambda^T(x) - f(x) \right\} \tag{1.5}$$

The framework of convex duality applies to convex functions as well, with minimum operator replaced by the maximum operator.

A disadvantage with factorized variational methods is that the quality of the approximation may not be necessarily high, depending on the dependency between hidden variables. Note that, aside from the two briefly sketched here, there exist other variational approaches for approximate Bayesian inference. These include the Bethe and Kikuchi family of variational methods (Yedidia, Freeman & Weiss 2001), approximations

to bound partition functions with higher order polynomials (Leisink & Kappen 2001), more elaborate variational methods for upper bounds on partition functions (Wainwright, Jaakkola & Willsky 2002).

1.2.3 Expectation-Propagation

EP is an iterative algorithm that tries to choose the best approximation of the posterior, within some tractable class of distributions. It is an extension of assumed-density filtering (ADF), e.g., (Maybeck 1979), which is a one-pass sequential method for computing an approximate posterior distribution. ADF processes observations sequentially, one by one, updating the posterior distribution after a data sample is observed. The sequential nature of ADF means that information that is discarded early one may turn out to be important at a later step. In contrast, EP makes additional passes, including information from later observations so that choices made earlier can be refined. This iterative refinement means that the resulting approximated posterior is independent of observation order and more accurate than ADF.

EP exploits the fact that the posterior can be expressed as a product of simpler parametric terms (i.e., in the exponential family). These simpler terms are approximated such that the Kulback-Leibler (KL) divergence (Kulback & Leibler 1951) between the true posterior and the approximated posterior are minimized. The result is a system of coupled equations that are iterated to reach a fixed-point. While EP has no issues with local minima and is typically faster than other approaches, it is not guaranteed to converge.

1.2.4 Laplace Approximations

Laplace's method approximates the posterior by a Gaussian distribution, which is found using a second order Taylor expansion around the model. Since the approximation is symmetric and locally fitted around the mode, the tails of the true posterior may not be covered, giving a poor approximation.

1.2.5 MCMC Sampling

MCMC methods generate samples from the posterior $p(\theta|D)$ by using evaluations of the unnormalized posterior $p(D|\theta)p(\theta)$. The statistics of the samples are used to approximate properties of the posterior distribution. A Markov chain of parameter values $\theta_0, \theta_1, ..., \theta_n$ is generated such that the distribution of θ_n asymptotically reaches that of the true posterior as the sequence length n increases.

A difficulty with MCMC sampling methods is determining how many steps are needed in order to converge to the desired distribution, i.e., the Markov chain's mixing time. However, MCMC sampling methods are advantageous when the posterior has a complex shape that is not necessarily Gaussian. Popular MCMC sampling methods include the Metropolis-Hastings method (Metropolis, Rosenbluth, Rosenbluth, Teller & Teller 1953, Hastings 1970), Gibbs sampling (Geman & Geman 1984), importance sampling (Robert & Casella 2005) and slice sampling (Neal 2003). More details on MCMC sampling methods can be found in (Neal 1993) and (MacKay 2003).

1.3 Key Challenges

In this dissertation, we consider supervised learning problems and are interested in scenarios with many data samples, high-dimensional data and where real-time learning (and fast computation) is needed. In particular, we focus on regression problems and learning models. Typical machine learning domains, in contrast, deal with offline analysis of training data and do not need any "online" learning, i.e., models are not updated or re-learnt during test time.

Challenges to autonomous real-time learning can be divided into modeling challenges and algorithmic constraints. The existence of noise in inputs, outliers, irrelevant and redundant variables, and high dimensions pose modeling challenges for algorithms. Additionally, we would like algorithms to satisfy the following algorithmic constraints: be runnable in real-time, computationally efficient, and autonomous (requiring no cross-validation or tuning of parameters). There exist a wealth of methods in the fields of machine learning and statistics that address some—bust not all–of these challenges and least of all not in an "autonomous" way.

For the rest of the manuscript, we will be putting the pieces of the puzzle together, introducing individual autonomous algorithms that progressively build on each other and that address larger subsets of challenges. Leveraging these autonomous methods, we will describe an automatic kernel shaping algorithm that learns the region of data samples that a local model should consider and cover. The algorithm can be applied not only local methods but global methods.

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows:

- Chapter 2 discusses how to perform linear regression in high-dimensional domains, where the input data has a large number of input dimensions—many of which are redundant or irrelevant.

- Chapter 3 considers the problem of high-dimensional linear regression when the input data is contaminated with noise. In such situations, typical regression methods are unable to handle noise in the input data and, consequently, produce biased estimates.

- Chapter 4 considers more realistic sensory data where outliers, as well as noise, are present in observed data. We propose a Bayesian formulation of weighted least squares that can automatically detect outliers in real-time. We incorporate this idea into the Kalman filter in order to learn an outlier-robust filter.

- Chapter 5 moves on to the nonlinear high-dimensional regression problem. We introduce a Bayesian kernel shaping algorithm that automatically learns the bandwidth of a local model. We demonstrate that Bayesian kernel shaping can be leveraged in not only local methods (such as locally weighted regression) but also in global methods that are linear in the parameters (such as Gaussian process regression).

- Chapter 6 concludes with a summary of contributions presented in this thesis and a discussion of future work.

Chapter 2

High Dimensional Linear Regression

In recent years, there has been growing interest in large scale analyses of brain activity with respect to associated behavioral variables. In the area of brain-machine interfaces, neural firing has been used to control an artificial system like a robot (Nicolelis 2001, Taylor, Tillery & Schwartz 2002), to control a cursor on a computer screen via non-invasive brain signals (Wolpaw & McFarland 2004), or to classify visual stimuli presented to a subject (Kamitani & Tong 2004, Haynes & Rees 2005). The brain signals are typically high dimensional, with large numbers of redundant and irrelevant signals. Linear modeling techniques like linear regression are among the primary analysis tools (Wessberg & Nicolelis 2004, Musallam, Corneil, Greger, Scherberger & Andersen 2004) for such data. 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. Good generalization is crucial for predicting behavior from future neural recordings. Two examples where accurate behavior prediction is relevant include i) the continual online interpretation of brain activity to control prosthetic devices and ii) 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 numerical problems is high. Classical techniques like ridge regression, stepwise regression (Draper & Smith 1981) or Partial Least Squares (PLS) regression (Wold 1975) are prone to overfitting and require careful human supervision for useful results.

Other methods such as Least Absolute Shrinkage and Selection Operator (LASSO) regression (Tibshirani 1996) attempt to shrink certain regression coefficients to zero by L1-norm regularization, resulting in interpretable models that are sparse. However, these L1-regularized regression methods have typically an open parameter, such as a regularization parameter, that needs to be set or optimized. Some of the methods for solving L1-regularized regression problems include convex optimization techniques such as sequential quadratic programming or interior-point methods, e.g., (Kim, Koh, Lustig, Boyd & Gorinevsky 2007), coordinate descent methods (Friedman, Hastie & Tibshirani 2007), the Gauss Seidel method (Shevade & Keerthi 2003), generalized iterative scaling (Goodman 2004), and iterative re-weighted least squares (Lokhorst 1999, Lee, Lee, Abeel & Ng 2006).

In this chapter, we focus on improving linear data analysis for the high dimensional scenarios described above. Our goal is to develop a statistically robust "black-box" approach that automatically detects the most relevant features for generalization. With the help of a variational Bayesian approximation and the introduction of "probabilistic backfitting" for linear regression, we develop a computationally efficient Bayesian treatment of linear regression with automatic relevance detection (ARD) (Neal 1994).

We demonstrate that the algorithm, called Variational Bayesian least squares (VBLS), can significantly improve the computational efficiency of sparse Bayesian learning, while performing feature detection and automatic relevance determination. Additionally, VBLS avoids any potentially expensive cross-validation or tuning of meta parameters by the user, offering a statistically robust, automatic method that can be applied across data sets from various systems.

VBLS can be interpreted as a Bayesian version of backfitting that does not require any sampling, making it suitable for implementation in incremental form for real-time applications (e.g., as in application domains such as robotics, brain-machine interfaces, tracking systems etc.). The algorithm's iterative nature is invaluable in real-time situations where decisions need to be made quickly such that an approximate solution is acceptable. In these scenarios, waiting a longer time for a very accurate solution may not be an acceptable alternative. The algorithm is most advantageous when embedded in other non iterative methods—such as the Relevance Vector Machine of Tipping (2001)—since it is able to offer significant relative computational improvement. In this way, we can apply the algorithm to high-dimensional problems in both linear and nonlinear scenarios.

We also present an incremental, real-time version of the algorithm, demonstrating its suitability for real-time interfaces between brains and machines. In addition to evaluations on several synthetic data sets and benchmark data sets, we apply the algorithm to the reconstruction of electromyographic (EMG) data from motor cortical firing, for the purpose of identifying if M1 and PM cortical neurons can directly predict EMG traces (Todorov 2000).

2.1 Computationally Tractable Linear Regression

Before developing our 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 $\mathbf{x}_i \in \Re^{d\times 1}$ are inputs and y_i are scalar outputs), the standard model for linear regression is:

$$y_i = \sum_{m=1}^{d} b_m x_{im} + \epsilon \tag{2.1}$$

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. Given D, the goal is to estimate the optimal linear coefficients $\mathbf{b} = [b_1 \; b_2 \; \cdots \; b_d]^T$ which combine the input dimensions to produce the output y.

It is easy to see that under our current noise model, the optimal estimate of the regression parameters (in a least-squares or maximum-likelihood sense) is given by:

$$\mathbf{b}_{\text{OLS}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{2.2}$$

where $\mathbf{X}$ denotes a matrix whose rows contain the vector $\mathbf{x}_i$ and $\mathbf{y}$ is a column vector containing the corresponding y_i. Eq. (2.2) is also known as the ordinary least squares (OLS) solution. The main problem with OLS regression in high-dimensional input spaces is that the full rank assumption of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ 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 $\left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)^{-1}$, 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, it 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) and Factor Regression (FR) (Massey 1965) are useful. These methods retain components in input space with large variance, regardless of whether these components 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 are projection regression techniques, most notably 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 high-dimensional, ill-conditioned regression problems, although it also has a tendency to overfit (Schaal et al. 1998).

There are also more efficient methods for matrix inversion, e.g., (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. Other sparse matrix factorization and conjugate gradient techniques, e.g., (Golub & Van Loan 1989, Chow &

Saad 1998, Smola & Scholkoph 2000), are preconditioners, attempting to transform matrices by factoring them into more computationally convenient and analytically simpler forms before inverting them.

Finally, there is a class of sparsity inducing methods such as LASSO (Least Absolute Shrinkage and Selection Operator) regression (Tibshirani 1996) that—along with other L1-regularized regression methods—attempt to shrink certain regression coefficients in the solution to zero by using a L1 penalty norm, instead of a 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). In these methods, the regularization parameter needs to be optimized, using cross-validation, convex optimization or other efficient search techniques.

In the next section, we describe a variational Bayesian linear regression algorithm (VBLS) that *automatically* regularizes against problems of overfitting. The iterative nature of the algorithm—due to its formulation as an EM problem—avoids the computational cost and numerical problems of matrix inversions that are faced in high-dimensional OLS regression (e.g., previously proposed sparse variational linear regression methods of Tipping (2001) and in Bishop (2006)). VBLS can be embedded into other iterative methods in order to realize computationally efficient updates, as we shall demonstrate in proceeding chapters.

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.

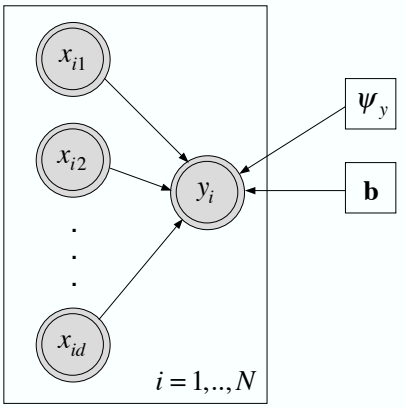


Figure 2.1: Graphical model for linear regression. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes.

Conceptually, the algorithm can be interpreted as a Bayesian version of either backfitting or PLS regression.

2.2 Variational Bayesian Least Squares

2.2.1 EM-based Linear Regression

Figures 2.1 to 2.3 illustrate the progression of graphical models needed to develop a robust Bayesian version of linear regression (D'Souza, Vijayakumar & Schaal 2004, Ting, D'Souza, Yamamoto, Yoshioka, Hoffman, Kakei, Sergio, Kalaska, Kawato, Strick & Schaal 2005).

Figure 2.1 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 "subset" selection techniques in regression (Wessberg & Nicolelis 2004). If we knew the optimal projection

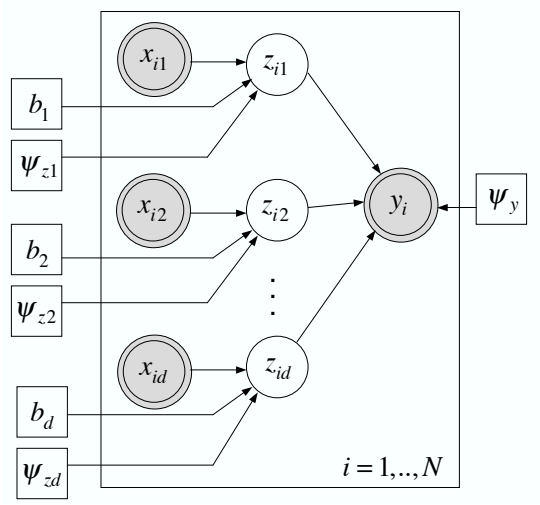


Figure 2.2: Graphical model for probabilistic backfitting. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes.

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 EM algorithms (Dempster et al. 1977). Figure 2.2 shows the corresponding graphical model.

The unobserved variables z_{im} (where $i = 1, \ldots, N$ 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. (2.1) to become:

$$y_i = \sum_{m=1}^{d} z_{im} + \epsilon_y \tag{2.3}$$

$$z_{im} = b_m x_{im} + \epsilon_{z_m} \tag{2.4}$$

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

$$\begin{aligned} y_i|\mathbf{z}_i &\sim \text{Normal}\left(y_i; \mathbf{1}^T\mathbf{z}_i, \psi_y\right) \\ z_{im}|x_{im} &\sim \text{Normal}\left(z_{im}; b_m x_{im}, \psi_{zm}\right) \end{aligned} \tag{2.5}$$

where $\mathbf{1} = [1, 1, \ldots, 1]^T$. While this model is still identical to OLS, notice that in the graphical model of Figure 2.2, the regression coefficients b_m are behind the fan-in to the outputs y_i. We call this model Probabilistic Backfitting since we can view the resulting derived update equation for the regression coefficient b_m as a probabilistic version of backfitting.

With the introduction of unobserved, random variables z_{im}, we are essentially in a situation where we wish to optimize the parameters $\phi = \left\{\{b_m, \psi_{zm}\}_{m=1}^{d}, \psi_y\right\}$, given that we have observed variables $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$. This situation fits very naturally into the framework of maximum-likelihood estimation via the EM algorithm.

The EM algorithm maximizes the *incomplete* log likelihood $\log p(\mathbf{y}|\mathbf{X})$ by maximizing the *expected complete* log likelihood $\langle \log p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \phi) \rangle$[1], where:

$$\begin{aligned}\log p(\mathbf{y}, \mathbf{Z}|\mathbf{X}; \phi) &= \log \left[\prod_{i=1}^{N} p(y_i|\mathbf{z}_i)(\mathbf{z}_i|\mathbf{x}_i) \right] = \sum_{i=1}^{N} \log p(y_i|\mathbf{z}_i) + \sum_{i=1}^{N} \log p(\mathbf{z}_i|\mathbf{x}_i) \\ &= -\frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^{N} \left(y_i - \mathbf{1}^T \mathbf{z}_i \right)^2 \\ &\quad - \frac{N}{2} \sum_{m=1}^{d} \log \psi_{zm} - \sum_{m=1}^{d} \frac{1}{2\psi_{zm}} \sum_{i=1}^{N} (z_{im} - b_m x_{im})^2 + \text{const}_{\mathbf{y},\mathbf{Z}} \end{aligned} \tag{2.6}$$

where $\mathbf{z}_i \in \Re^{d \times 1}$ consists of z_{im} elements and $\mathbf{Z} \in \Re^{N \times d}$ consists of the vectors z_i in its rows. The resulting EM updates require standard manipulations of normal distributions—please refer to appendix B.4 of (D'Souza 2004) for the derivations—and are shown below:

E-step :

$$\mathbf{1}^T \mathbf{\Sigma}_z \mathbf{1} = \left(\sum_{m=1}^{d} \psi_{zm} \right) \left[1 - \frac{1}{s} \left(\sum_{m=1}^{d} \psi_{zm} \right) \right] \tag{2.7}$$

$$\sigma_{zm}^2 = \psi_{zm} \left(1 - \frac{1}{s} \psi_{zm} \right) \tag{2.8}$$

$$\langle z_{im} \rangle = b_m x_{im} + \frac{1}{s} \psi_{xm} \left(y_i - \mathbf{b}^T \mathbf{x}_i \right) \tag{2.9}$$

M-step :

$$b_m = \frac{\sum_{i=1}^{N} \langle z_{im} \rangle \, x_{im}}{\sum_{i=1}^{N} x_{im}^2} \tag{2.10}$$

$$\psi_y = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \mathbf{1}^T \langle \mathbf{z}_i \rangle \right)^2 + \mathbf{1}^T \mathbf{\Sigma}_z \mathbf{1} \tag{2.11}$$

$$\psi_{zm} = \frac{1}{N} \sum_{i=1}^{N} \left(\langle z_{im} \rangle - b_m x_{im} \right)^2 + \sigma_{zm}^2 \tag{2.12}$$

[1] where $\langle \cdot \rangle$ denotes the expectation operator

where we define $s = \psi_y + \sum_{m=1}^{d} \psi_{xm}$ and $\mathbf{\Sigma}_z = \text{Cov}(\mathbf{z}|\mathbf{y}, \mathbf{X})$ is the covariance matrix of $\mathbf{z}$. This EM version of least squares regression is guaranteed to converge to the same solution as OLS (D'Souza et al. 2004).

One EM update has a computationally complexity of $O(Nd)$ *per EM iteration*, where d is the number of input dimensions, instead of the $O(d^3)$ associated with OLS regression or even $O(d^2)$, if more efficient and robust matrix inversion methods are used. This efficiency comes at the cost of an iterative solution, instead of a one-shot solution for $\mathbf{b}$ as in OLS. Should the number of EM iterations be significant, it is true that the run-time of the EM algorithm could be as long as non-iterative approaches. However, as previously mentioned, the true benefit of our iterative approach arises when dealing real-time applications (where decisions need to be made quickly in a short amount of time such that an approximate solution is acceptable) and also when embedded in other iterative methods in order to realize more computationally efficient update equations.

The new EM algorithm appears to only replace the matrix inversion in OLS by an iterative method, as others have done with alternative algorithms (Strassen 1969, Hastie & Tibshirani 1990). 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.

2.2.2 Automatic Relevance Determination

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

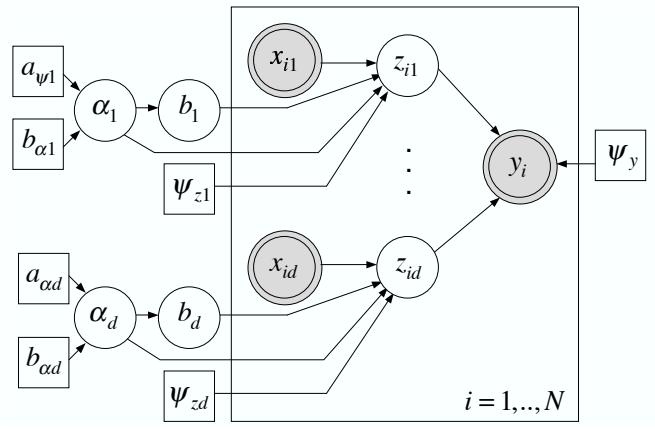


Figure 2.3: Graphical model for variational Bayesian least squares. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes.

purpose, as shown in Figure 2.3, we introduce precision variables α_m over each regression parameter b_m, as previously done in (Tipping 2001):

$$\begin{aligned} \mathbf{b}|\boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Normal}\left(b_m; 0, 1/\alpha_m\right) \\ \boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Gamma}\left(\alpha_m; a_{\alpha_m,0}, b_{\alpha_m,0}\right) \end{aligned} \tag{2.13}$$

where $\boldsymbol{\alpha} \in \Re^{d\times 1}$ consists of α_m components and $\{a_{\alpha_m,0}, b_{\alpha_m,0}\}$ are the initial hyperparameter values for α_m.

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 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. (2.13) looks very similar to that of Tipping (2001) and the later work of Bishop (2006), our model has the key property that it is computationally efficient, requiring $O(Nd)$ per EM iteration. In contrast, the methods of Bishop (2006) and Tipping (2001) take $O(d^3)$ and $O(N^3)$, per iteration, respectively, becoming prohibitively expensive for large data sets with a very large input dimensionality, d. It is the 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:

$$\begin{aligned} y_i|\mathbf{z}_i &\sim \text{Normal}\left(\mathbf{1}^T\mathbf{z}_i, \psi_y\right) \\ z_{im}|b_m, \alpha_m, x_{im} &\sim \text{Normal}\left(b_m x_{im}, \frac{\psi_{zm}}{\alpha_m}\right) \\ b_m|\alpha_m &\sim \text{Normal}\left(0, \frac{1}{\alpha_m}\right) \\ \alpha_m &\sim \text{Gamma}(a_{\alpha_m,0}, b_{\alpha_m,0}) \end{aligned} \tag{2.14}$$

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.

The dependency of z_{im} on the precision α_m may seem unnecessary, but Gelman, Carlin, Stern & Rubin (2000) provide a justification: it is reasonable to assume that the variance in z_{im} scales with the variance in b_m since increasing our uncertainty in the prior of b_m should imply a corresponding increase in the uncertainty of z_{im} as well. In this case, we will obtain a joint posterior distribution $Q(\mathbf{b}, \alpha)$, which is then marginalized to get the individual distributions $Q(\mathbf{b})$ and $Q(\alpha)$. With this formulation, the marginal distribution over $\mathbf{b}$ is now a product of Student-t distributions instead of the Gaussian distributions.

As the graphical model in Figure 2.3 shows, the set of unobserved variables in the model is now $\left\{\mathbf{b}, \boldsymbol{\alpha}, \{\mathbf{z}_i\}_{i=1}^N\right\}$. 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 $\langle \log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha}|\mathbf{X}; \phi)\rangle$:

$$
\begin{aligned}
&\log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha}|\mathbf{X}; \phi) \\
&= \log p(\mathbf{y}, \mathbf{Z}, \mathbf{b}, \boldsymbol{\alpha}|\mathbf{X}; \boldsymbol{\Psi}_z, \psi_y, a_{\boldsymbol{\alpha}}, b_{\boldsymbol{\alpha}}) \\
&= \log \left[\prod_{i=1}^{N} p(y_i|\mathbf{z}_i) p(\mathbf{z}_i|\mathbf{b}, \boldsymbol{\alpha}, \mathbf{x}_i) p(\mathbf{b}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha})\right] \\
&= \sum_{i=1}^{N} \log p(y_i|\mathbf{z}_i) + \sum_{i=1}^{N}\sum_{m=1}^{d} \log p(z_{im}|b_m, \alpha_m, x_{im}) + \sum_{m=1}^{d} \log p(b_m|\alpha_m) + \sum_{m=1}^{d} \log p(\alpha_m)
\end{aligned}
$$

$$
\begin{aligned}
&= -\frac{N}{2}\log\psi_y - \frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i - \mathbf{1}^T\mathbf{z}_i\right)^2 \\
&\quad - \frac{N}{2}\sum_{m=1}^{d}\log\frac{\psi_{zm}}{\alpha_m} - \sum_{m=1}^{d}\frac{\alpha_m}{2\psi_{zm}}\left(z_{im} - b_m x_{im}\right)^2 \\
&\quad + \frac{1}{2}\sum_{m=1}^{d}\log\alpha_m - \frac{1}{2}\sum_{m=1}^{d}\alpha_m b_m^2 \\
&\quad + \sum_{m=1}^{d}\left(a_{\alpha_m,0} - 1\right)\log\alpha_m - \sum_{m=1}^{d} b_{\alpha_m,0}\alpha_m + \text{const}_{\mathbf{y},\mathbf{Z},\mathbf{b},\boldsymbol{\alpha}}
\end{aligned} \tag{2.15}
$$

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 Figure 2.3 is an intentional design. As previously mentioned, under this graphical model, the marginal distribution of b_m becomes a Student t-distribution, allowing for traditional hypothesis testing (Gelman et al. 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 (Jordan et al. 1999, Jaakkola & Jordan 2000, Attias 2000, Ghahramani & Beal 2000). We will return to this point in the Discussion section.

After some algebraic manipulations (details on the derivations can be found in Appendix A.1), the final EM posterior update equations become:

E-step:

$$\boldsymbol{\Sigma}_{\mathbf{z}} = \left(\frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \boldsymbol{\Psi}_z^{-1}\langle\mathbf{A}\rangle\right)^{-1} = \boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1} - \frac{\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}\mathbf{1}^T\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}}{\psi_y + \mathbf{1}^T\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}} \tag{2.16}$$

$$\begin{aligned}\langle\mathbf{z}_i\rangle &= \boldsymbol{\Sigma}_{\mathbf{z}}\left(\frac{1}{\psi_y}\mathbf{1}y_i + \boldsymbol{\Psi}_z^{-1}\langle\mathbf{A}\rangle\langle\mathbf{B}|\mathbf{A}\rangle\,\mathbf{x}_i\right) \\ &= \left(\frac{\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}}{\psi_y + \mathbf{1}^T\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}}\right)y_i + \left(\langle\mathbf{B}|\mathbf{A}\rangle - \frac{\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}\mathbf{1}^T\langle\mathbf{B}|\mathbf{A}\rangle}{\psi_y + \mathbf{1}^T\boldsymbol{\Psi}_z\langle\mathbf{A}\rangle^{-1}\mathbf{1}}\right)\mathbf{x}_i\end{aligned} \tag{2.17}$$

$$\sigma^2_{b_m|\alpha_m} = \frac{\psi_{zm}}{\langle\alpha_m\rangle}\left(\sum_{i=1}^N x_{im}^2 + \psi_{zm}\right)^{-1} \tag{2.18}$$

$$\langle b_m|\alpha_m\rangle = \left(\sum_{i=1}^N x_{im}^2 + \psi_{zm}\right)^{-1}\left(\sum_{i=1}^N \langle z_{im}\rangle\, x_{im}\right) \tag{2.19}$$

$$\hat{a}_{\alpha_m} = a_{\alpha_m,0} + \frac{N}{2} \tag{2.20}$$

$$\hat{b}_{\alpha_m} = b_{\alpha_m,0} + \frac{1}{2\psi_{zm}}\left\{\sum_{i=1}^N \langle z_{im}^2\rangle - \left(\sum_{i=1}^N x_{im}^2 + \psi_{zm}\right)^{-1}\left(\sum_{i=1}^N \langle z_{im}\rangle\, x_{im}\right)^2\right\} \tag{2.21}$$

$$\langle\alpha_m\rangle = \frac{\hat{a}_{\alpha m}}{\hat{b}_{\alpha m}} \tag{2.22}$$

M-step:

$$\psi_y = \frac{1}{N}\sum_{i=1}^N \left(y_i - \mathbf{1}^T\langle\mathbf{z}_i\rangle\right)^2 + \mathbf{1}^T\boldsymbol{\Sigma}_z\mathbf{1} \tag{2.23}$$

$$\psi_{zm} = \frac{1}{N}\sum_{i=1}^{N} \langle \alpha_m \rangle \left(\langle z_{im} \rangle - \langle b_m | \alpha_m \rangle \, x_{im} \right)^2 + \langle \alpha_m \rangle \, \sigma_{zm}^2 + \langle \alpha_m \rangle \, \sigma_{b_m|\alpha_m}^2 \left(\frac{1}{N}\sum_{i=1}^{N} x_{im}^2 \right) \tag{2.24}$$

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. $\boldsymbol{\Sigma}_\mathbf{z}$ is a diagonal covariance matrix with a diagonal vector of σ_z^2. Note that:

$$\left\langle z_{im}^2 \right\rangle = \langle z_{im} \rangle^2 + \sigma_{z_m}^2$$

where $\sigma_{z_m}^2$ is the mth term of the vector σ_z^2.

Initialization of Priors: The hyperparameters of α_m are learnt using EM, as shown by Eqs. (2.20) and (2.21). We set the initial values of the hyperparameters, $a_{\boldsymbol{\alpha},0}$ and $b_{\boldsymbol{\alpha},0}$, in an uninformative way and use values of $a_{\alpha_m,0} = 10^{-8}$ and $b_{\alpha_m,0} = 10^{-8}$ for all $m = 1, \ldots, d$. This means that initial value of α_m is 1, with high uncertainty, i.e., α_m has a rather flat prior distribution. These initial hyperparameter values for α_m need never be changed, regardless of the data set or system. In this way, the algorithm retains a "black-box" like quality.

Notice that the update equation for $\langle b_m | \alpha_m \rangle$ can be rewritten recursively so that the posterior mean of b_m in the $(n+1)$th EM iteration is:

$$\langle b_m | \alpha_m \rangle^{(n+1)} = \left(\frac{\sum_{i=1}^{N} x_{im}^2}{\sum_{i=1}^{N} x_{im}^2 + \psi_{zm}} \right) \langle b_m | \alpha_m \rangle^{(n)} + \frac{\psi_{zm}}{s\alpha_m} \frac{\sum_{i=1}^{N} \left(y_i - \langle \mathbf{b}|\boldsymbol{\alpha} \rangle^{(n)T} \mathbf{x}_i \right) x_{im}}{\sum_{i=1}^{N} x_{im}^2 + \psi_{zm}} \tag{2.25}$$

where $s = \psi_y + \mathbf{1}^T \mathbf{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}$. Examining Eq. (2.25), we see that the first time is a decaying term. In the absence of a correlation between the current input dimension and the residual error (i.e., if the second term is zero), then after some number of EM iterations, the mean of the current regression coefficient $\langle b_m | \alpha_m \rangle$ will approach zero. 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, Eqs. (2.16) to (2.24), have an algorithmic complexity of $O(Nd)$ per EM iteration. However, the number of EM iterations required before convergence is an open issue and could be many. Hence, VBLS is most advantageous in incremental, real-time scenarios where, due to hard time constraints, an approximately accurate solution is satisfactory in lieu of an accurate solution that takes unacceptably long to compute. 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 $2\hat{a}_\alpha$ degrees of freedom, which allows a principled way of determining whether a regression coefficient was excluded by means of standard hypothesis testing.

The pseudocode for VBLS can be found in Appendix A.2.

2.3 Real-time Implementation

Due to its computationally efficient nature, the VBLS algorithm presented in Algorithm 1 lends itself to scenarios where fast, online learning with large amounts of high-dimensional data is required, such as real-time brain-machine interfaces. Previous work by Sato &

Ishii (2000) and Sato (2001) has shown that an online version of the Variational Bayes framework can be derived, such that online model selection can be done with guaranteed convergence. A scalar discount factor or forgetting rate is typically introduced in order to forget estimates that were calculated earlier (and hence, were less accurate). (Sato & Ishii 2000, Sato 2001) introduce a time-dependent schedule for the discount factor and prove convergence of the online EM-based algorithm. Since the main focus of this chapter is on the batch form of the algorithm, we will show only a proof-of-concept and use a constant-valued discount factor (with a heuristically-set value) in order to demonstrate that the batch VBLS algorithm can be translated into incremental form. We leave the detailed theoretical development of the online version of the algorithm with a discount factor schedule for future work.

In particular, we introduce a forgetting rate, $0 \leq \lambda \leq 1$, to exponentially discount data collected in the past, as done in (Ljung & Soderstrom 1983). The forgetting rate enters the algorithm by accumulating sufficient statistics of the batch algorithm in an incremental way. Setting $\lambda = 0$ ensures that all past samples are forgotten, while setting $\lambda = 1$ ensures that none of observed samples are forgotten. We can then extract the sufficient statistics by examining the batch EM equations, Eqs. (2.16) to (2.24). The incremental EM update equations are listed in Appendix A.3.

2.4 Evaluation

We turn to the application and evaluation of VBLS in the context of predicting EMG data from neural data recorded in the primary motor (M1) and premotor (PM) cortices of

monkeys (Ting et al. 2005, Ting, D'Souza, Yamamoto, Yoshioka, Hoffman, Kakei, Sergio, Kalaska, Kawato, Strick & Schaal 2008). 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.

2.4.1 Synthetic Data

2.4.1.1 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 $\mathbf{b} \in \Re^{10\times 1}$, 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:

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

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 $\sigma_{noise}^2 = c\sigma_{\bar{y}}^2$, 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) $v = 60, u = 30$

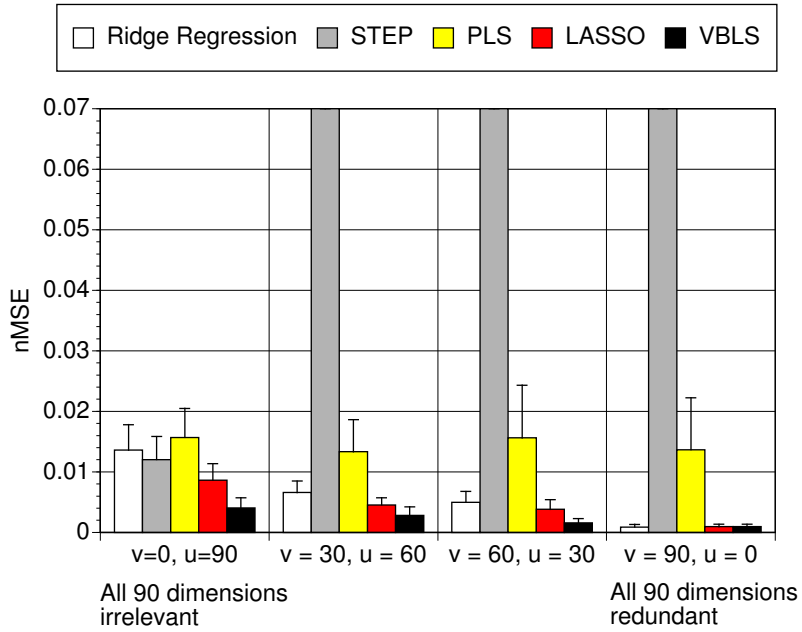
iv) $v = 90, u = 0$ (all the 90 input dimensions are redundant)

2.4.1.2 Methods

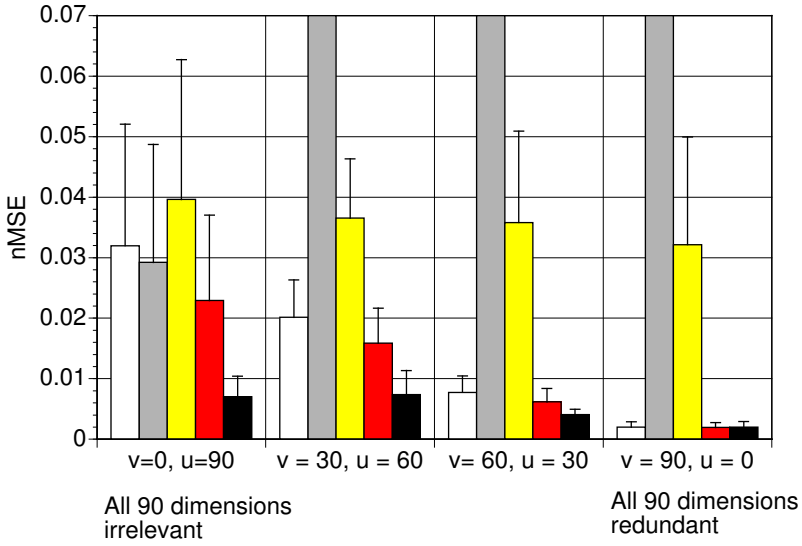
We compared VBLS to four other methods that were previously described in Section 2.1: i) ridge regression, ii) stepwise regression, iii) PLS regression and iv) LASSO regression. For ridge regression, we introduced a small ridge parameter value of 10^{-10} to avoid ill-conditioned matrix inversions. We 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.

2.4.1.3 Results

For evaluation, we calculated the prediction error on noiseless test data, using the learnt regression coefficients from each technique. Results are quantified as normalized mean squared errors (nMSE). Figure 2.4 shows the average prediction error for noiseless test data, given training data where the output noise is either high ($r^2 = 0.8$) or low(er) ($r^2 = 0.9$).



(a) Average prediction error for training data where output noise has $r^2 = 0.9$.



(b) Average prediction error for training data where output noise has $r^2 = 0.8$.

Figure 2.4: 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. The number of redundant dimensions is denoted by v, and the number of irrelevant dimensions is u.

All the algorithms were executed on 10 randomly generated sets of data. The predictive nMSE results reported in Figure 2.4 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 Figures 2.4(a) and 2.4(b), we see that regardless of output noise level, VBLS achieves either the lowest predictive nMSE 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 Figures 2.4(a) and 2.4(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.

2.4.1.4 Non-Normal Synthetic Data

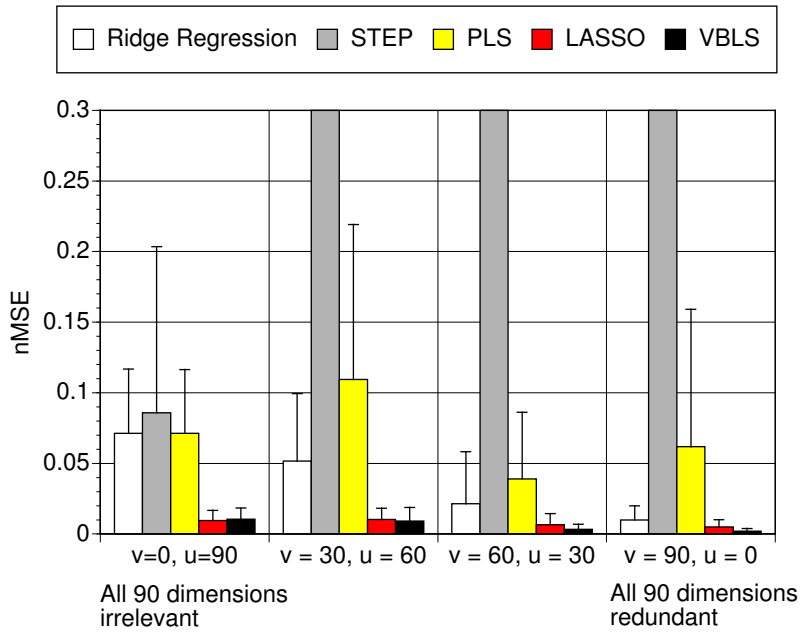


Figure 2.5: Average normalized mean squared prediction error for synthetic non-Normal 100 input-dimensional data, with an output noise of $r^2 = 0.9999$ in the training data, averaged over 10 trials. The number of redundant dimensions is denoted by v, and the number of irrelevant dimensions is u.

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 2.4.1.1, with 100 dimensions—10 of which are relevant dimensions. The other 90 dimensions are chosen to be either irrelevant or redundant. However, there are two differences between this synthetic data and that of Section 2.4.1.1.

Firstly, 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 $m = 1, ..., 10$ 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 the non-Normal synthetic data set and the data of Section 2.4.1.1 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 2.4.1.1. The test data was created in a similar manner, except the input and output data were left noise-free. As in Section 2.4.1.1, we considered synthetic training data with $N = 1000$ data samples and a synthetic test data set with 20 data samples.

Figure 2.5 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.

2.4.2 EMG Prediction from Neural Firing

2.4.2.1 Data sets

We investigated data from two different neurophysiological experiments. In the first experiment by Sergio & Kalaska (1998), a monkey moved a manipulandum in a center-out task in eight different directions, equally spaced in a horizontal planar circle of 8cm 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[2].

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 10msec) and the corresponding EMG activation[3] from a particular muscle. A sampling interval of 10msec was used. For each sample in this data set, a delay of 50msec between M1 cortical neural firing and EMG muscle activation was empirically chosen, based on estimates from measurements.

The second experiment, conducted by Kakei, Hoffman & Strick (1999, 2001), involved a monkey trained to perform eight different combinations of wrist flexion-extension and

[2]The 11 arm muscles analyzed included the 1) surpraspinatus, 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.

[3]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 10msec bins) to generate summed histograms of activity. The EMG data had no physically meaningful units.

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 data set (Kakei et al. 1999), the EMG outputs of 7 contributing muscles[4] 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. Similar to the Sergio & Kalaska (1998) data set, each data sample consisted of the average firing rates from a particular neuron (averaged over a window of 10msec) and the corresponding EMG activation from a particular muscle. A sampling interval of 10msec was used. For each sample in the Kakei et al. (1999) data set, a delay of 20msec[5] between M1 cortical neural firing and EMG muscle activation was chosen empirically, based on estimates from measurements. The second data set (Kakei et al. 2001) 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, the second Kakei et al. (2001) data set had 2592 data samples for each neuron/muscle pair. For each sample, a delay of 30msec[6] between PM cortical neural firing and EMG muscle activation was assumed.

[4]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.

[5]The results of our analyses are insensitive to a delay in the range of $20 - 60$msec since there was only a very small numerical difference between the quality of the fit of the data in this interval. Delays of 50msec or higher are physiologically more plausible.

[6]Within a delay range of $30 - 80$msec, there is no real difference in the quality of fit of our analyses.

2.4.2.2 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 (1998) data, given 71 neurons, the number of possible models that exist for a particular muscle is:

$$\sum_{m=1}^{71} \begin{pmatrix} 71 \\ m \end{pmatrix}$$

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 (1998) 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. experiments, 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.

Figure B.1 in Appendix B.1 shows how these 8 cross-validation sets are constructed from the Sergio & Kalaska (1998) 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 2.4.1.2, 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 Figure 2.6 demonstrate how well each algorithm performs, averaging the generalization performances over all the cross-validation sets from Figure B.1.

The average number of relevant neurons[7] (i.e., not including irrelevant neurons and neurons providing redundant information), shown in Figure 2.9, was calculated by averaging over the number of relevant neurons in each of the 8 training sets in Figure B.1.

The final set of relevant neurons, used in Figure B.4 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

[7]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

reported in Figure B.4 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. (1999) M1 neural and Kakei et al. (2001) PM neural data sets 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. Figures B.2 and B.3 in Appendix B.1i 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 (1998) data set. The average normalized mean squared error values shown in Figures 2.7 and 2.8 illustrate the generalization performance of each algorithm, averaged over all the cross-validation sets shown in Figures B.2 and B.3[8]. The average number of relevant neurons shown in Figures 2.10 and 2.11 was calculated by averaging over the number of relevant neurons found in each of the 6 training sets from Figures B.2 and B.3.

[8]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.

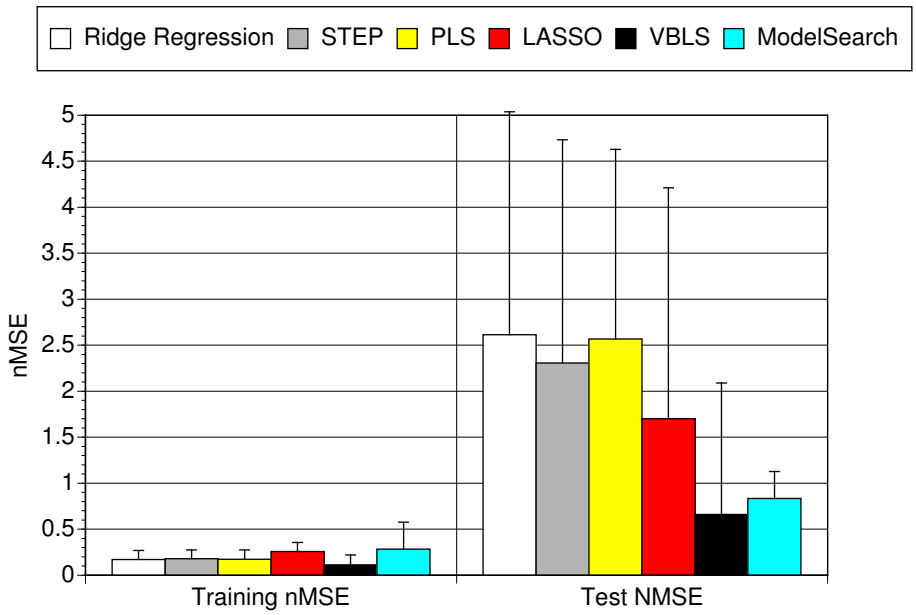


Figure 2.6: Normalized mean squared error, averaged over all cross-validation sets and over all muscles for Sergio & Kalaska (1998) M1 neural data set.

As with the Sergio & Kalaska (1998) data set, the final set of relevant neurons—used in Figures 2.10 and 2.11—was obtained for each algorithm (except VBLS) by taking the common neurons found to be relevant over the 6 cross-validation sets.

2.4.2.3 Results

Generalization Performance: Figures 2.6 to 2.8 show that VBLS resulted in a generalization error comparable to that produced by the baseline study. In the Kakei et al. (1999) M1 and Kakei et al. (2001) PM neural datasets, all algorithms performed similarly. However, ridge regression, stepwise regression, PLS regression and LASSO regression performed far worse on the Sergio & Kalaska (1998) M1 neural dataset, with ridge regression attaining the worst error. Such performance is typical for traditional linear regression methods on ill-conditioned high-dimensional data, motivating the development of VBLS.

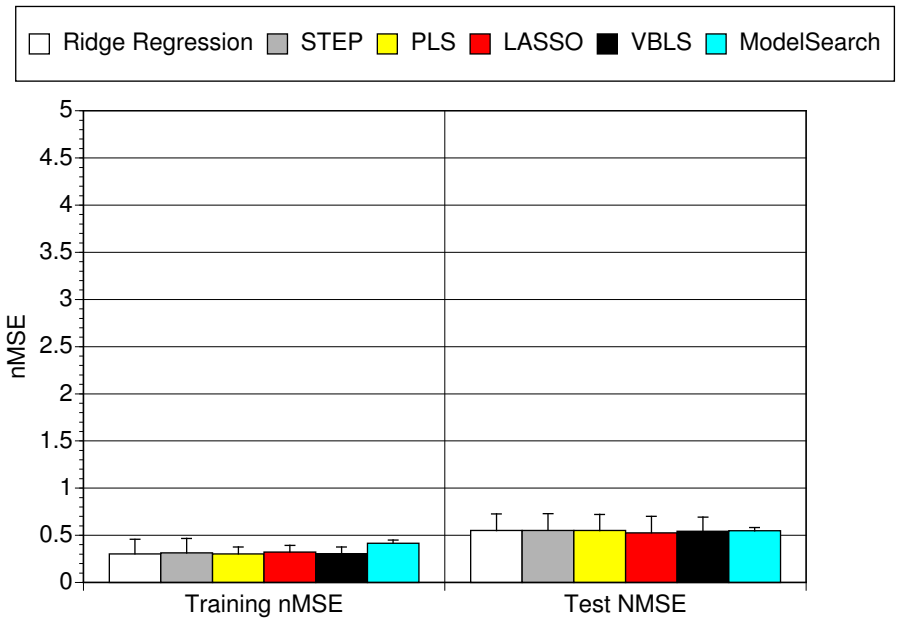


Figure 2.7: Normalized mean squared error, averaged over all cross-validation sets and over all muscles for Kakei et al. (1999) M1 neural data.

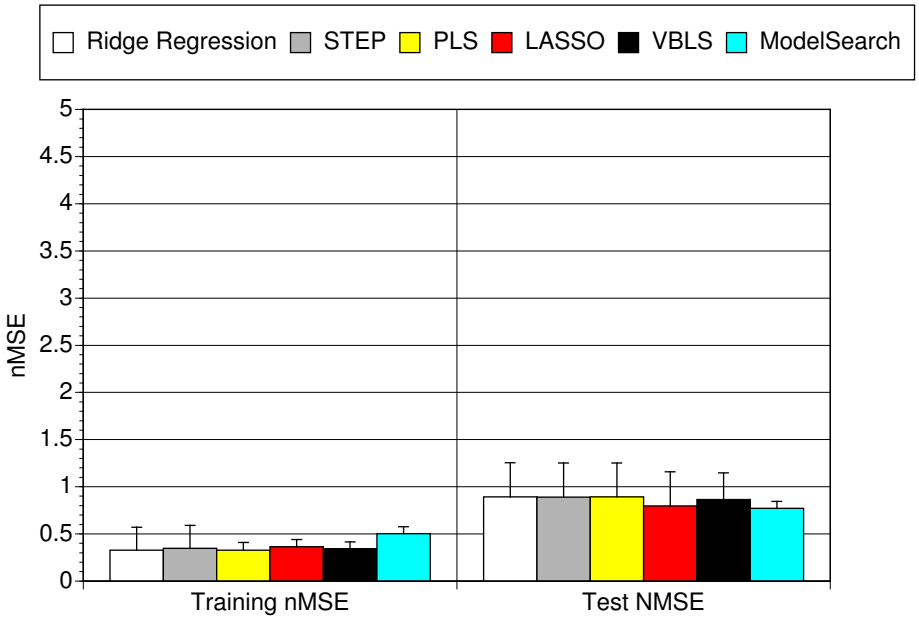


Figure 2.8: Normalized mean squared error, averaged over all cross-validation sets and over all muscles for Kakei et al. (2001) PM neural data.

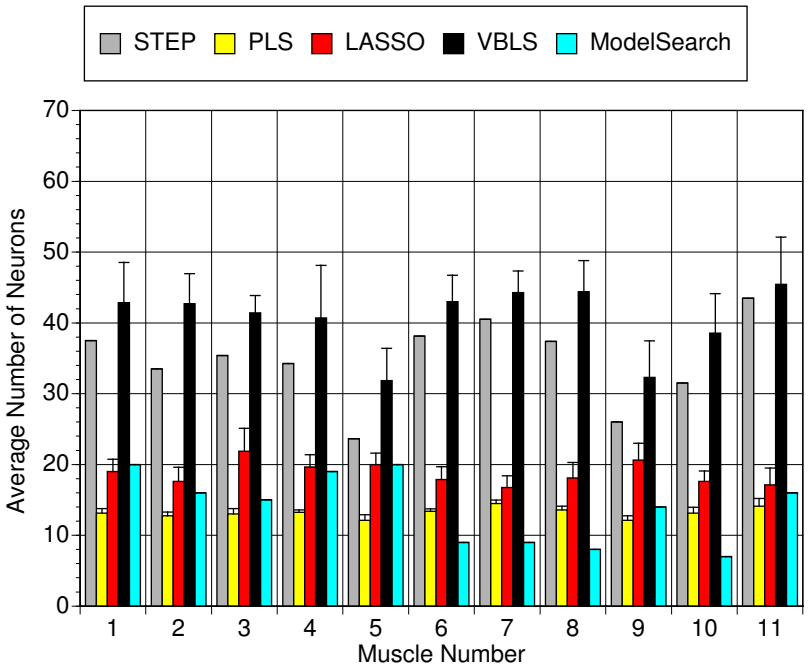


Figure 2.9: Average number of relevant M1 neurons found over all cross-validation sets for Sergio & Kalaska (1998) data set.

Interestingly, in Figure 2.7, 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. (1999) 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. (2001) PM neural data set when examining Figure 2.8, although this effect is less pronounced in the PM neural data set. In contrast, Figure 2.6 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 (1998) M1 neural data set is somehow much richer and hence, more challenging to analyze.

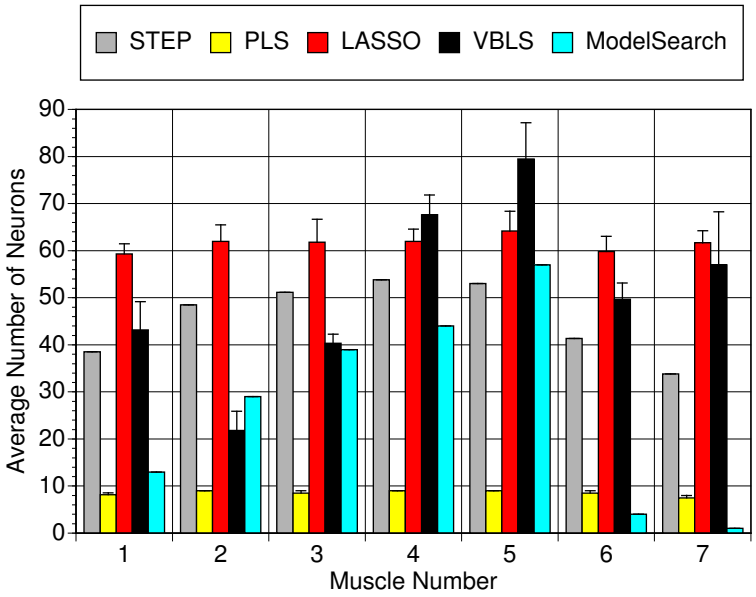


Figure 2.10: Average number of relevant M1 neurons found over all cross-validation sets for Kakei et al. (1999) data.

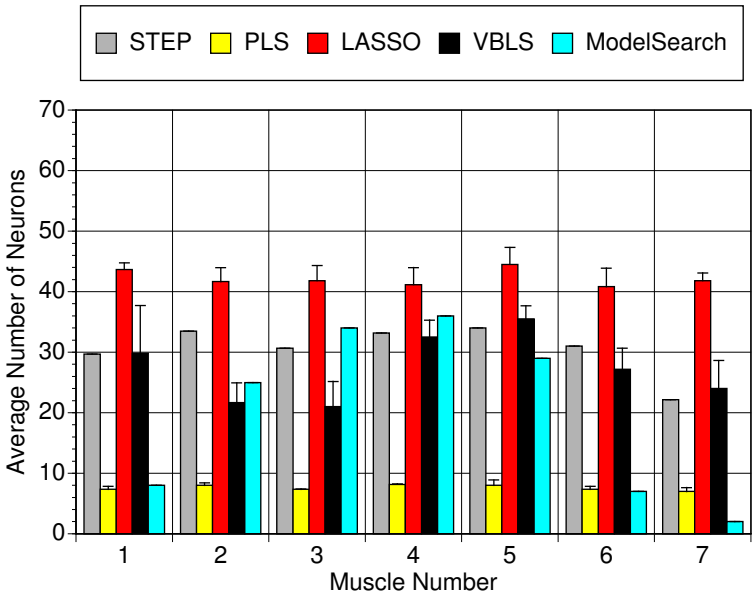


Figure 2.11: Average number of relevant PM neurons found over all cross-validation sets for Kakei et al. (2001) data.

Table 2.1: Percentage neuron matches found by each algorithm, as compared to those found by the baseline study (ModelSearch), averaged over all muscles of each data set.

	STEP	**PLS**	**LASSO**	**VBLS**
Sergio & Kalaska (1998) M1 data	7.2%	7.4%	5.4%	94.2%
Kakei et al. (1999) M1 data	65.1%	42.9%	80.6%	94.4%
Kakei et al. (2001) PM data	22.9%	14.2%	44.5%	91.5%

Average Number of Relevant Neurons: The average number of relevant M1 neurons found by VBLS was slightly higher than the baseline study, as seen in Figures 2.9 to 2.11. This is unsurprising since the baseline studies did not consider all possible combination of neurons. For example, the baseline study for Sergio & Kalaska (1998) data set considered possible combinations of up to only 20 neurons, instead of the full set of 71 neurons. In particular, notice that in Figures 2.10 and 2.11, 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).

Percentage Relevant Neuron Match: 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 in comparison to the model search approach. Table 2.1 shows that the relevant neurons identified by VBLS coincided at a very high percentage with those of the baseline model search results, while stepwise and PLS regression had inferior outcomes. The 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 hours 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 sets, but fared far worse on the Sergio & Kalaska (1998) 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.

Figures B.4, B.5 and B.6 in Appendix B.2 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 shown in Figures 2.6, 2.7 and 2.8 suggests that the Bayesian approach of VBLS sufficiently regularizes the participating neurons such that no overfitting occurs, despite finding a larger number of relevant neurons.

In general, VBLS achieved comparable performance with the baseline study when reconstructing EMG data from M1 or PM neurons. Note that VBLS is an iterative statistical method, which performs slower than classical "one-shot" linear least squares methods (i.e., on the order of several minutes for the data sets in our analyses). Nevertheless, it achieves comparable results with our combinatorial model search, while performing at *much faster speeds*.

2.4.3 Real-time Analysis for Brain-Machine Interfaces

Both neural data sets analyzed in Section 2.4.2 are inherently real-time data—collected online, stored and then analyzed in batch form (i.e., a sampling interval is used, and a delay between neural firing and EMG activity is empirically chosen in order to extract the data samples to be used in the batch form of the data). As a result, in the real-time

simulations, we took the batch form of the data and presented it sequentially, one data sample at each time step.

We applied the real-time version of VBLS, derived in Section 2.3, on the Sergio & Kalaska (1998) data set since this was the more interesting of the three presented in Section 2.4.2. We used a forgetting rate of $\lambda = 0.999$, assumed each sample of the data set arrived sequentially at different time steps, and iterated through the incremental VBLS equations, Eqs. (A.13) to (A.12), twice for each time step.

Figure B.7(a) shows the coefficient of determination values, r^2 (where $r^2 = 1 - \text{nMSE}$), for both the batch and real-time versions of VBLS on the entire Sergio & Kalaska (1998) data set. Figure B.7(b) shows the number of relevant M1 neurons found by batch VBLS and real-time VBLS for the same data set. For the real-time version of VBLS, the r^2 values and relevant neurons reported were from the last time step. We can see from both figures that the real-time and batch versions of VBLS achieve a similar level of performance. The average r^2 values—averaged over all 11 muscles—confirm this: batch VBLS had an average r^2 value of 0.7998, while real-time VBLS had an average r^2 value of 0.7966.

2.5 Interpretation of Neural Data Analyses

While the main focus of this chapter lies in the introduction of a robust linear regression technique for high-dimensional data, we would like to discuss how our analysis technique can be exploited for the interpretation of the neurophysiological data that we used in this

study. We show plots of EMG activity for a muscle in the Sergio & Kalaska (1998) data set, but further plots referred to in the text below can be found in Appendix B.4.

Each of the eight EMG plots in Figures 2.12 and 2.13 shows the following three EMG traces:

i) the raw average EMG trajectories

ii) the predicted EMG activity, as obtained by VBLS using all available data in all conditions (VBLS-full)

iii) the predicted EMG activity, as obtained by VBLS using only half the data for fitting[9] (VBLS-cv)

This last cross-validated fit tests how well isometric M1 neural recordings can predict movement EMG and how well movement-related M1 neural recordings can predict isometric EMG. Alternatively, it tests whether the neuron to EMG relationship is the same between the isometric and the movement conditions.

2.5.1 Sergio & Kalaska (1998) data set

One of the main results reported by Sergio & Kalaska (1998) was that the firing of the reported M1 neurons had strong correlation with EMG-like (or force-like) signals in both movement and isometric conditions. In contrast, evidence for correlations with kinematic data (such as movement direction, velocity, or target direction) was less pronounced.

We generated Figures 2.12 and 2.13, both of which reproduce similar illustrations to Figures 3A and 3B in Sergio & Kalaska (1998). The two figures show the EMG activity

[9]For the isometric condition, only movement data was used for fitting. For the movement condition, only isometric data was used for fitting.

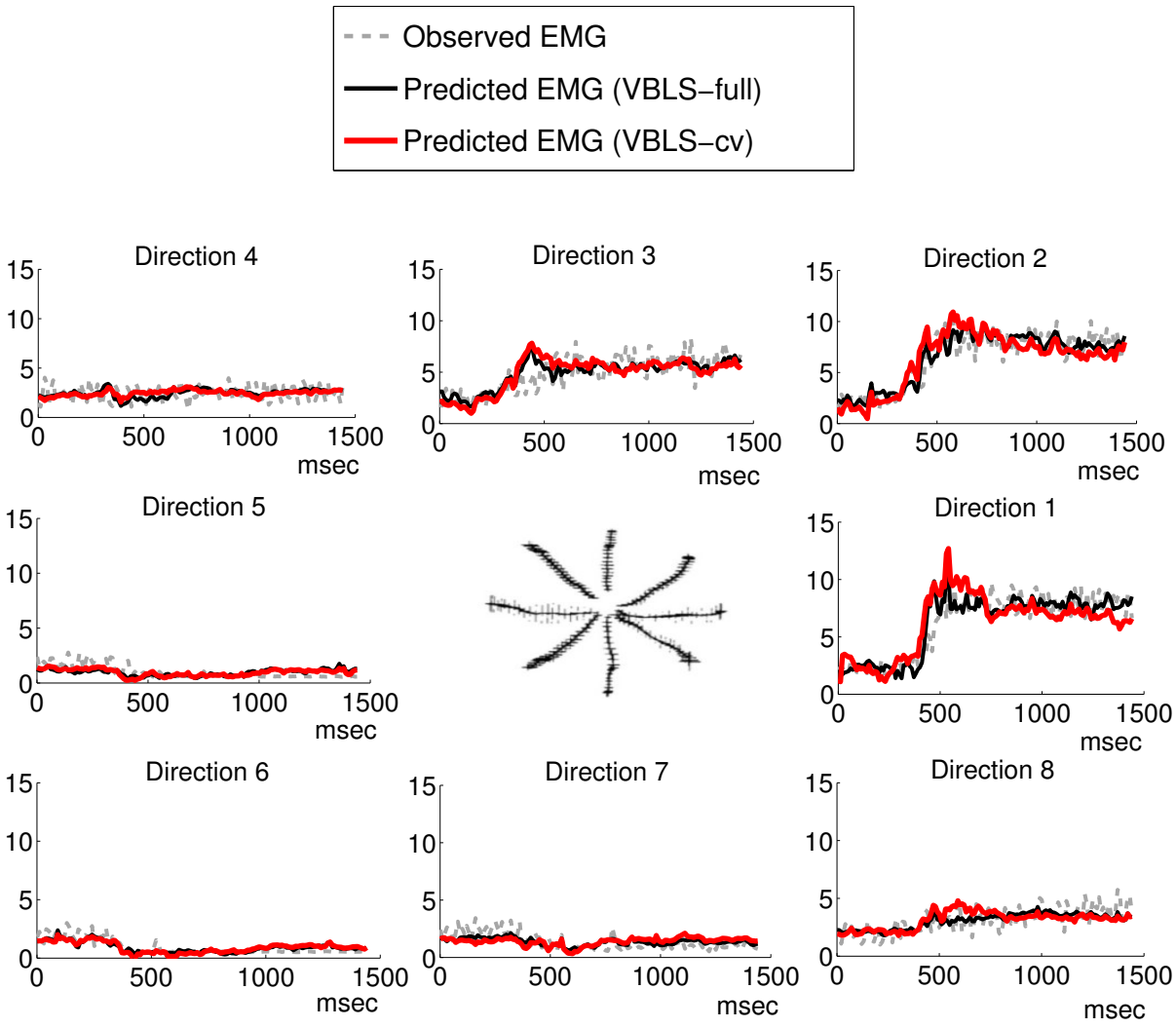


Figure 2.12: Observed vs. predicted EMG traces under isometric force conditions for the infraspinatus muscle, given M1 neural firing from Sergio & Kalaska (1998). The center plot shows the trajectories in eight different directions—in the (x, y) plane—taken by the hand. This figure is taken from Sergio & Kalaska (1998). Each of the eight plots surrounding this center plot shows EMG traces over time for each hand trajectory.

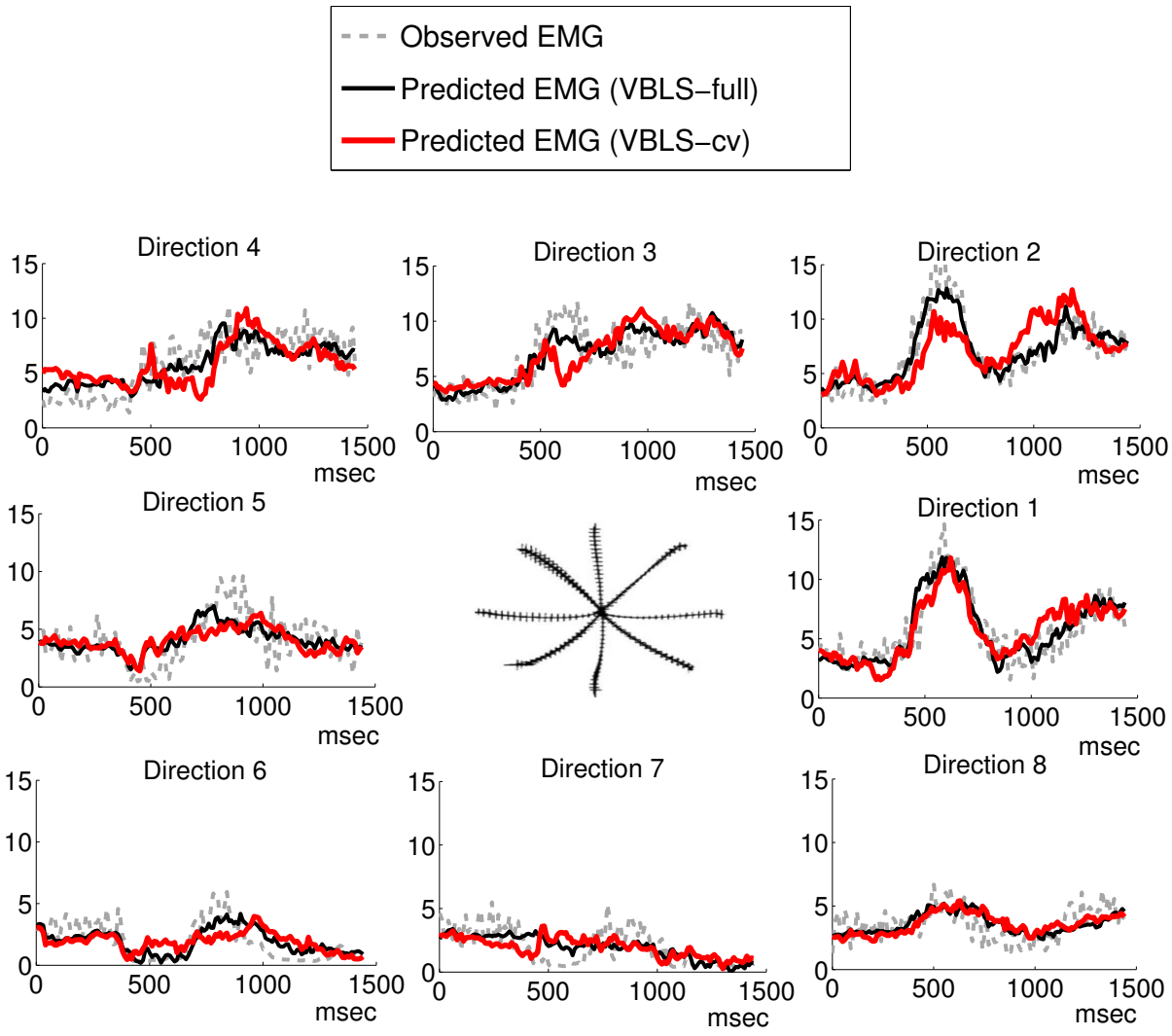


Figure 2.13: Observed vs. predicted EMG traces under movement force conditions for the infraspinatus muscle, given M1 neural firing from Sergio & Kalaska (1998). The center plot shows the trajectories in eight different directions—in the (x, y) plane—taken by the hand. This figure is taken from Sergio & Kalaska (1998). Each of the eight plots surrounding this center plot shows the EMG traces over time for each hand trajectory.

of the infraspinatus muscle in all eight isometric force production directions (Figure 2.12) and movement directions (Figure 2.13). The trajectories, shown in (x, y) coordinates, taken by the hand are illustrated in the center of each figure. These center figures are taken from the original figures of Sergio & Kalaska (1998) since we did not have access to the hand trajectory data.

As Figures 2.12 and 2.13 both show, M1 neural firing predicts the EMG traces very well in general. The cross-validation tests also demonstrate very good EMG reconstruction, *thus confirming Sergio & Kalaska (1998)'s results that the recorded M1 neurons have sufficient information to extract signals of the time-varying dynamics and the temporal envelopes of EMG activities.*

2.5.2 Kakei et al. (1999) and Kakei et al. (2001) data sets

The main message in Kakei et al. (1999) and Kakei et al. (2001) was that one can find neurons in M1 that carry intrinsic (muscle-based) information and neurons that carry extrinsic–that is, (x, y) task space–information. In contrast, the PM cortex had predominantly extrinsic neurons.

For our analyses, we had access to the average firing rates of the M1 and PM neurons and the corresponding EMG traces, as well as the (x, y) movement as performed by the hand. Thus, we used VBLS to predict the EMG activity in all three arm posture conditions (pronated, supinated and midway between the two) from the neural firing and to predict the (x, y)-velocity trajectories from neural firing. Note that all this data was obtained from the same highly trained monkey, such that it was possible to i) re-use

EMG data obtained during the M1 experiment as target for the PM data and ii) share the same (x, y) data across the M1 and PM experiment.

We illustrate our results in a similar form as in Figures 2.12 and 2.13, showing plots for the extensor carpi radialis brevis (ECRB) muscle and only for the supination posture. Figure B.8(a) shows the EMG fits for M1 neurons, while Figure B.9(a) shows the same fits for PM neurons. The center plots illustrate recorded (x, y) movement in the horizontal plane in this posture. Interestingly, **both M1 and PM neurons achieve a very good EMG reconstruction**[10].

Figures B.10(a), B.12(a), B.11(a) and B.13(a) demonstrate the (x, y)-velocity fits for M1 and PM neurons, respectively, in the supination condition[11]. The quality of fit appears reduced in comparison to the EMG data, but it is hard to quantify this statement as EMG and (x, y)-velocities have quite different noise levels such that r^2 values cannot be compared.

In order to judge whether M1 or PM neurons achieve better fits for EMG and (x, y)-velocity data, we compared the r^2 values from all experimental conditions in a pairwise student's t-test. No significant difference could be found between either the quality of EMG fitting or the (x, y)-velocity fits.

Thus, our analysis concludes that **both** M1 and PM carry sufficient information to predict EMG activity. It should be noted, however, that in Kakei et al.'s original

[10]It should be noted that, potentially, the hand movement from Kakei et al. is of significant lower complexity than the arm movement data of Sergio & Kalaska (1998). The temporal profiles of the EMG data in Kakei et al. is much simpler, such that it may be easier to predict it. Support for this latter hypothesis comes from the fact that essentially all statistical methods we tested performed equally well on the EMG prediction problem. Thus, future work will have to examine whether PM neurons would also be able to predict more complex EMG traces.

[11]The optimal delay value between M1 cortical neural firing and the resulting direction of movement was found to be 80msec since this value lead to the lowest fitting error. In a similar fashion, the optimal delay between PM cortical neural firing and the resulting direction of movement was found to be 90msec.

experiment, neurons were classified into extrinsic or intrinsic neurons according to how much their tuning properties were compatible with intrinsic or extrinsic variables. Their analysis was a single neuron analysis, while our investigation looked at the predictive capabilities of the entire population of neurons. Thus, our results are not in contradiction with Kakei et al., but rather, demonstrate the important difference between the predictive capabilities of a single neuron vs. that of the population code. The latter is of particular importance for brain-machine interfaces, and our results provide further evidence for the information richness of cortical areas that, from the view of single neuron analysis, seemed to be much more specialized.

We also analyzed the neurons that were found to be relevant for EMG prediction and (x, y)-velocity prediction, using t-tests performed on the inferred regression coefficients. In particular, we wondered whether some neurons in PM and M1 would specialize on EMG prediction, while others would prefer (x, y)-velocity prediction. However, no interesting specialization could be found. For example, of all 72 PM neurons, we found that 4.17% were relevant to (x, y)-velocity prediction only, 15.28% were relevant to EMG prediction only, and 79.17% were relevant to both velocity and EMG prediction (leaving 1.39% of PM neurons to be irrelevant to both velocity and EMG prediction). Of all 92 M1 neurons, we found that 4.35% were relevant to (x, y)-velocity prediction only, 26.09% were relevant to EMG prediction only, and 65.22% were relevant to both velocity and EMG prediction. Thus, the majority of neurons were involved in both EMG and velocity prediction.

This rich information about different movement variables in both M1 and PM most likely contributes to the success of various brain-machine interface projects, where the precise placement of electrode arrays seemingly does not matter too much.

2.6 Discussion

This chapter addresses the problem of analyzing high-dimensional data with linear regression techniques, typically encountered in neuroscience and the new field of brain-machine interfaces. In order to achieve robust statistical results, we introduced a novel Bayesian technique for linear regression analysis with automatic relevance determination, called Variational Bayesian Least Squares. In contrast to previously proposed variational linear regression methods, VBLS is computationally efficient, requiring $O(d)$—instead of $O(d^3)$—updates per EM iteration. It is guaranteed to converge to the global solution. VBLS has no parameters that need to be tuned; the prior distribution of $\boldsymbol{\alpha}$ can be set to be wide and uninformative (e.g., the hyperparameters $a_{\alpha,0}$ and $b_{\alpha,0}$ can be set to 10^{-8}) and *need never be changed from data set to data set*, making it "black-box"-like and autonomous in its execution.

While VBLS is an iterative statistical method (performing slower than classical one-shot linear least squares), it can be embedded into other more complex iterative methods to realize a savings in computational efficiency. Its iterative nature means that it is most advantageous in real-time scenarios where time constraints favor an approximately accurate solution that is computed quickly over an extremely accurate solution that takes unacceptably too long. VBLS also lends itself to incremental implementation as would be needed in real-time analyses of brain information.

A point of concern that one could raise against the VBLS algorithm is in how far the variational approximation in this algorithm affects the quality of function approximation. It is known that factorial approximations to a joint distribution create more peaked

distributions, such that one could potentially assume the VBLS might tend a bit towards overfitting. It is important to notice, however, that in the case of VBLS, a more peaked distribution over the posterior distribution of b_m actually entails a stronger bias towards excluding the associated input dimensions. A more peaked distribution over b_m pushes the regression parameter closer to zero. Thus, VBLS will be on the slightly pessimistic side of function fitting and is unlikely to overfit, which corresponds to our empirical experience.

EMG Prediction from Neural Firing: Our final application of VBLS examined how well motor cortical activity can predict EMG activity and end-effector velocity data as collected in monkey experiments in previous publications (Sergio & Kalaska 1998, Kakei et al. 1999, Kakei et al. 2001). Our analysis confirmed that neurons in M1 carry significant information about EMG activity and end-effector velocity. These results were also obtained in the original papers but with single-neuron analysis techniques and not a population code read-out as essentially performed by VBLS. Interestingly, we also discovered that PM carries excellent information about EMG and end-effector velocity—it has been previously suggested that only end-effector information is the primary variable coded in PM. Most likely, this result is due to using population code-based analysis instead of single neuron analysis.

Our findings did not suggest that either M1 or PM has a significant specialized population of neurons that only correlates with either EMG or end-effector data. Instead, we found that most neurons were statistically significant for both EMG and end-effector

data prediction. This rich information in the motor cortices mostly likely contributes significantly to the success of brain-machine interface experiments, where electrode arrays are placed over large cortical areas and the reconstruction of behavioral variables seems to be relatively easy. VBLS offers an interesting new method to perform such read-outs even in real-time with high statistical robustness.

Chapter 3

Parameter Identification in Noisy Linear Regression

Learning the equations of motion of a complex physical system for the purpose of control is a common problem in robotics. A typical system identification approach would first collect a representative data set from the robot by measuring positions and motor commands during some explorative movements. Then, velocity and acceleration information would be obtained by numerical differentiation of position data. The data would also be digitally filtered to reduce noise. As a third step, a function approximator would be applied to learn the mapping from positions, velocities and accelerations to motor commands. Such a function often has hundreds of inputs for complex robots. Finally, this mapping would be inserted into the control loop of the robot, where appropriate motor commands are predicted from desired position, velocity and acceleration information—all of which are noiseless data.

The example scenario above is representative for a large number of system identification problems. From a machine learning point of view, the interesting components are that the learning data is high dimensional, has irrelevant and redundant dimensions and, despite digital filtering, usually contains a significant amount of noise in the inputs to

the function approximator. Moreover, predictions are required from *noiseless* input data since inputs generated during control originate from a planning system without noise. The quality of control strongly depends on the quality of the learnt internal model in advanced controllers and is critical in many robotic applications such as haptic devices, surgical robotics and safe compliant assistive robots in human environments.

Ideally, system identification can be performed based on the CAD data of a robot provided by the manufacturer, at least in the context of rigid body dynamic (RBD) systems—which will be the exemplary scope of this paper. However, many modern light-weight robots such as humanoid robots have significant additional nonlinear dynamics beyond the rigid body dynamics model, due to actuator dynamics, routing of cables, use of protective shells and other sources. In such cases, instead of trying to explicitly model all possible nonlinear effects in the robot, empirical data-driven system identification methods appear to be more useful. Under the assumption that a rigid body dynamics model is sufficient to capture the entire robot dynamics, this problem is theoretically straightforward as all unknown parameters of the robot such as mass, center of mass and inertial parameters appear linearly in the rigid body dynamics equations (An, Atkeson & Hollerbach 1988). Hence, after appropriate re-arrangement of the RBD equations of motion, parameter identification can be performed with linear regression techniques.

In this chapter, we address the problem above in the context of linear regression since an extension to nonlinear regression is straightforward using locally weighted learning methods (Atkeson, Moore & Schaal 1997). If we wanted to use traditional linear regression techniques for this scenario, we would encounter several deficiencies.

Ill-conditioned data: For high dimensional robotic systems, it is not easy to generate sufficiently rich data so that all parameters will be properly identifiable. As a result, the regression problem for RBD parameter estimation is almost always numerically ill-conditioned and bears the danger of generating parameter estimates that strongly deviate from the true values, despite a seemingly low error fit of the data. For such ill-conditioned data sets in high dimensional spaces, most traditional linear regression techniques break down numerically since they are unable to generate sparse and unbiased solutions identifying redundant and/or irrelevant dimensions.

Noisy sensory data: Observed sensory data is typically noisy. Noise sources exist in both input and output data, and this effect is additionally amplified by numerical differentiation to obtain derivative data. Even digital filtering will always leave some noise in the signal in order to avoid oversmoothing of data. Traditional linear regression techniques like OLS regression are only capable of dealing with noise in the output data, and the presence of input noise introduces a persistent bias to the regression solution. Alternative methods such as Total Least Squares (TLS) (Golub & Van Loan 1989, Van Huffel & Vanderwalle 1991)—otherwise known as orthogonal-least squares regression (Hollerbach & Wampler 1996) or, in statistics, as errors-in-variables (EIV) when applied to a linear model (Van Huffel & Lemmerling 2002)—address input noise, but they assume the variances of input noise and output noise are the same (Rao & Principe 2002). In real-world systems, this assumption is not necessarily true and, again, the resulting estimates will be biased, leading to inferior generalization. There also exist noise-robust versions of standard algorithms such

as Principal Component Analysis (PCA) (Sanguinetti, Milo, Rattray & Lawrence 2005) and other approaches such as Factor Analysis for regression, e.g., (Massey 1965), that attempt to explicitly model the input noise.

Physically implausible parameter estimates: Finally, there is no mechanism in the regression problem for RBD model identification that ensures the identified parameters are physically plausible. Particularly in the light of insufficiently rich data and nonlinearities beyond the RBD model, one often encounters physically incorrectly identified parameters such as negative values on the diagonal of an inertial matrix.

Various methods exist to deal with some of the problems mentioned above, such as regression based on singular-value decomposition (SVD) or ridge regression to cope with ill-conditioned data (Belsley, Kuh & Welsch 1980), stepwise regression (Draper & Smith 1981), LASSO regression (Tibshirani 1996) or other L1-regularized methods to produce sparse solutions, and TLS/orthogonal-least squares/EIV, Factor Analysis or noise-robust PCA to address input noise (Hollerbach & Wampler 1996). Nevertheless, a comprehensive approach *addressing the entire set of issues* has not been suggested so far. Recent work such as Rao, Erdogmus, Rao & Principe (2003) has addressed the problem of input noise, but in the context of system identification of a time-series, while ignoring the problems associated with ill-conditioned data in high dimensional spaces.

In this chapter, we motivate the problem of input noise in linear regression applications and identify possible solutions. Leveraging the Bayesian framework for linear regression developed in Chapter 2, we present an extension of the algorithm that can handle noisy

high-dimensional input data, while using Bayesian regularization methods to ensure robustness to ill-conditioned data (Ting, D'Souza & Schaal 2006). A post-processing step ensures that the rigid body parameters are physically consistent by nonlinearly projecting the results of the Bayesian estimate onto the constraints. We evaluate our approach on a parameter estimation problem for the RBD model (Ting, Mistry, Peters, Schaal & Nakanishi 2006).

Our Bayesian estimation approach to the RBD parameter estimation that has all the desired properties below:

- Explicitly identifies input and output noise in the data
- Is robust in face of ill-conditioned data
- Detects non-identifiable parameters
- Produces physically correct parameter estimates

3.1 Background

3.1.1 Parameter Estimation in Rigid Body Dynamics

Let us examine some of the problems associated with traditional system identification methods before introducing our de-noising solution. We embed our discussions in the context of RBD parameter estimation—a problem that is linear in the open parameters despite the high level of nonlinearity of the RBD equations of motion. We discuss general nonlinear system identification at the end of this chapter.

The general RBD equations of motions are (Sciavicco & Siciliano 1996):

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau \tag{3.1}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$ denote the vectors of joint positions, velocities, and accelerations, respectively. The matrix $\mathbf{M}(\mathbf{q})$ is the RBD inertial matrix, the matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ has terms about coriolis and centripetal forces, and the vector $\mathbf{G}(\mathbf{q})$ represents torques due to gravity. Eq. (3.1) has one row for every degree-of-freedom (DOF) of the robot, e.g., 30-50 rows for a humanoid robot. Every DOF is physically characterized by at least 10 parameters: a mass parameter, a center of mass vector, and a positive definite inertial matrix; friction parameters can increase the number of parameters. Thus, for robot systems with many DOFs, identifying RBD parameters is a problem involving hundreds of dimensions. Interestingly, these parameters appear linearly in Eq. (3.1), such that, after some complex rearrangement of the terms in Eq. (3.1), the system identification problem for RBD becomes a linear regression problem.

We can now switch to viewing this system identification problem from the stance of machine learning. Let us assume we have a data set $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ consisting of N samples, where $\mathbf{x}_i \in \Re^{d\times 1}$ (d is the dimensionality of the input data) and y_i is a scalar. As mentioned previously, the RBD equations can be re-arranged to yield this structure. We create a matrix $\mathbf{X} \in \Re^{N\times d}$, where the input vectors $\mathbf{x}_i$ are arranged in the rows of $\mathbf{X}$, and a vector $\mathbf{y} \in \Re^{N\times 1}$, where the corresponding scalar outputs y_i are coefficients of $\mathbf{y}$.

3.1.2 Modeling Input Noise in Linear Regression

In Chapter 2, we introduced a computationally efficient Bayesian linear regression algorithm that is suitable for high-dimensional data. Unfortunately, it does not model noise in input data. To address this, we express the general model for linear regression with noise-contaminated input and output data as follows:

$$\begin{aligned} y_i &= \sum_{m=1}^{d} w_{zm} t_{im} + \epsilon_{y_i} \\ x_{im} &= w_{xm} t_{im} + \epsilon_{x_{im}} \end{aligned} \tag{3.2}$$

where $\mathbf{t}_i$ is noiseless input data composed of t_{im} elements, $\mathbf{w}_z$ and $\mathbf{w}_x$ are regression vectors composed of w_{zm} and w_{xm} elements, respectively, and ϵ_y and ϵ_x are additive mean-zero Gaussian noise. Only $\mathbf{X}$ and $\mathbf{y}$ are observable.

If the input data is noiseless (i.e., $x_m = w_{xm} t_m$), then we obtain the familiar linear regression equation of $y = \mathbf{b}_{\text{OLS}}^T \mathbf{x} + \epsilon_y$, where $b_{\text{OLS},m} = w_{zm}/w_{xm}$. We can show this using simple algebraic manipulation:

Given:

$$y_i = \sum_{m=1}^{d} w_{zm} t_{im} + \epsilon_{y_i} \tag{3.3}$$

$$x_{im} = w_{xm} t_{im} \text{ or, equivalently, } t_{im} = \frac{x_{im}}{w_{xm}} \tag{3.4}$$

Substituting Eq. (3.4) into Eq. (3.3) gives: $y_i = \sum_{m=1}^{d} b_{\text{OLS},m} t_{im} + \epsilon_y$, where $b_{\text{OLS},m} = w_{zm}/w_{xm}$. The slightly more general formulation in Eq. (3.2) with distinct w_{xm} and w_{zm}

coefficients will be useful in preparing our new algorithm. All the linear methods reviewed in Chapter 2 do not account for noise in input data.

When the input data is contaminated with noise, it can be shown that the OLS estimate will be $\mathbf{b}_{\text{OLS,noise}} = \gamma \mathbf{b}_{\text{true}}$, where $0 < \gamma <= 1$ (and $\mathbf{b}_{\text{true}} = \mathbf{b}_{\text{OLS}}$), and the exact value of γ depends on the amount of input noise. Consider the predictive distribution $p(y^q|\mathbf{x}^q, \mathbf{X}, \mathbf{y}) = \int p(y^q, \mathbf{t}|\mathbf{x}^q, \mathbf{X}, \mathbf{y}) p(\mathbf{t}) d\mathbf{t}$. If we assume that the prior distributions on y and $\mathbf{t}$ are Gaussians, then we know that $p(y^q|\mathbf{x}^q, \mathbf{X}, \mathbf{y})$ is a Gaussian as well. We can then evaluate the integral and infer the mean and variance of $p(y^q|\mathbf{x}^q, \mathbf{X}, \mathbf{y})$, resulting in the following:

$$\begin{aligned} \langle y^q|\mathbf{x}^q \rangle &= \mathbf{W}_z \mathbf{W}_x \left(\mathbf{W}_x^T \mathbf{W}_x + \mathbf{\Psi}_x \right)^{-1} \mathbf{x}^q \\ &= \mathbf{b}_{\text{OLS,noise}} \mathbf{x}^q \end{aligned}$$

where $\mathbf{W}_z$ is a diagonal matrix with w_{zm} entries on its diagonal (and similarly, for $\mathbf{W}_x$ and $\mathbf{\Psi}_x$). Further algebraic manipulation of the expression for $\mathbf{b}_{\text{OLS,noise}}$ reveals the following:

$$\begin{aligned} \mathbf{b}_{\text{OLS,noise}} &= \mathbf{W}_z \mathbf{W}_x \left(\mathbf{W}_x^T \mathbf{W}_x + \mathbf{\Psi}_x \right)^{-1} \\ &= \mathbf{W}_z \mathbf{W}_x \mathbf{W}_x^{-1} \mathbf{W}_x^{-1} \left(\mathbf{W}_x^T \mathbf{W}_x \mathbf{W}_x^{-1} \mathbf{W}_x^{-1} + \mathbf{\Psi}_x \mathbf{W}_x^{-1} \mathbf{W}_x^{-1} \right)^{-1} \\ &= \mathbf{W}_z \mathbf{W}_x^{-1} \left(\mathbf{I} + \mathbf{\Psi}_x \mathbf{W}_x^{-1} \mathbf{W}_x^{-1} \right)^{-1} \\ &= \mathbf{b}_{\text{OLS}} \left(\mathbf{I} + \mathbf{\Psi}_x \mathbf{W}_x^{-1} \mathbf{W}_x^{-1} \right)^{-1} \end{aligned} \tag{3.5}$$

where, in the second line of Eq. (3.5), we introduce a multiplicative factor of $\mathbf{W}_x^{-1} \mathbf{W}_x^{-1} \mathbf{W}_x \mathbf{W}_x$ or $\mathbf{I}$ (the identity matrix). Hence, $\mathbf{b}_{\text{OLS,noise}}$ is always less than or equal to $\mathbf{b}_{\text{true}}$. Thus,

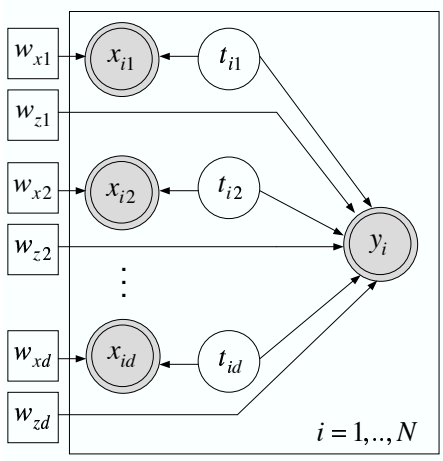


Figure 3.1: Graphical model for joint Factor Analysis for regression. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes.

OLS regression underestimates the regression vector and produces biased predictions, a problem that cannot be fixed by adding more training data.

Intentionally, the input/output noise model formulation in Eq. (3.2) was chosen such that it coincides with a version of a Factor Analysis (Massey 1965) tailored for regression problems (please refer to Appendix C.1 for more details on Factor Analysis for regression). The intuition of this model is given in Figure 3.1: every observed input x_{im} and output y_i is assumed to be generated by a set of hidden variables t_{im} and contaminated with some noise, as given in Eq. (3.2).

The graphical model in Figure 3.1 compactly describes the full multi-dimensional system: the variables x_{im}, t_{im}, w_{xm} and w_{zm} are duplicated d times for the d input dimensions of the data—as represented by the four nodes in the plate indexed by m. The other plate, indexed by i, shows that there are N samples of observed $\{\mathbf{x}_i, y_i\}$ data. The goal of learning is to find the parameters w_{xm} and w_{zm}, which can only be achieved

by estimating the hidden variables t_{im} and the variances of all random variables. In order to ensure that all parameters of the model are well-constrained, it needs to be assumed that all t_{im} follow a Gaussian distribution with mean zero and unit variance, i.e., $t_{im} \sim \text{Normal}(0, 1)$.

The specific version of Factor Analysis for regression depicted in Figure 3.1 is called joint-space Factor Analysis regression or Joint Factor Analysis (JFA) regression, as both input and output variables are treated the same in the estimation process (i.e., only their joint distribution matters). While Joint Factor Analysis regression is well-suited for modeling regression problems with noisy input data, it does not handle ill-conditioned data very well and is computationally expensive for high dimensions due to a repeated high dimensional matrix inversion in the ensuing iterative estimation procedure.

The goal of learning is to find parameters w_{xm} and w_{zm}, which can only be achieved by estimating the hidden variables t_{im}, z_{im} and the variances of all random variables. Optimal prediction can then be performed with either noisy or noiseless inputs, by deriving the appropriate conditional distributions.

In the following section, we develop a Bayesian treatment of Joint Factor Analysis regression that is robust to ill-conditioned data, automatically detects non-identifiable parameters, detects noise in input and output data.

3.2 Bayesian Regression with Input Noise

3.2.1 EM-based Joint Factor Analysis

To start with, we introduce the hidden variables z_{im}, as done in Section 2.2, so that our noisy linear regression model from Eq. (3.2) now becomes:

$$\begin{aligned} y_i &= \sum_{m=1}^{d} z_{im} + \epsilon_{y_i} \\ z_{im} &= w_{zm} t_{im} + \epsilon_{z_{im}} \\ x_{im} &= w_{xm} t_{im} + \epsilon_{x_{im}} \end{aligned} \tag{3.6}$$

The new model is shown graphically in Figure 3.2. We use the EM algorithm to determine all open parameters using maximum likelihood estimation, with the following probability distributions over random variables:

$$\begin{aligned} y_i|\mathbf{z}_i &\sim \text{Normal}(\mathbf{1}^T \mathbf{z}_i, \psi_y) \\ z_{im}|t_{im}, w_{zm} &\sim \text{Normal}(w_{zm} t_{im}, \psi_{zm}) \\ x_{im}|t_{im}, w_{xm} &\sim \text{Normal}(w_{xm} t_{im}, \psi_{xm}) \\ t_{im} &\sim \text{Normal}(0, 1) \end{aligned} \tag{3.7}$$

where $\mathbf{1} = [1, 1, ..., 1]^T$, $\mathbf{z}_i \in \Re^{d \times 1}$ is composed of z_{im} elements, $\mathbf{w}_z \in \Re^{d \times 1}$ is composed of w_{zm} elements, and $\mathbf{w}_x$, ψ_z and ψ_x are similarly composed of w_{xm}, ψ_{zm} and ψ_{xm} elements, respectively. As Figure 3.2 shows, the regression coefficients w_{zm} are now behind the fan-in to the output y_i.

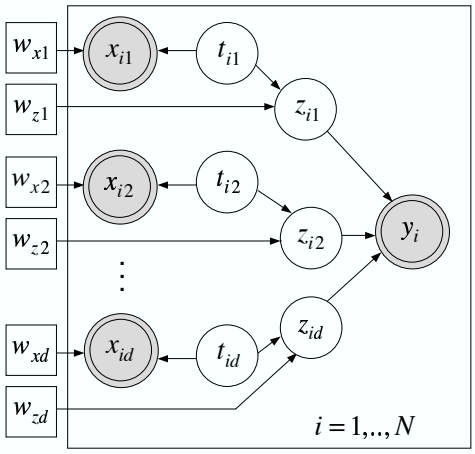


Figure 3.2: Graphical model for joint factor analysis for efficient estimation. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes. Note that square nodes for the noise variances ψ_y and ψ_{zm} have been omitted from the model for graphical clarity.

This efficient Joint Factor Analysis formulation decouples the input dimension and operates with $O(d)$ per EM iteration—where d is the number of input dimensions, instead of the approximately $O(d^3)$ per EM iteration in traditional Joint Factor Analysis.

3.2.2 Automatic Feature Detection

The efficient maximum likelihood formulation of Joint Factor Analysis regression is, however, still vulnerable to ill-conditioned data. Thus, we introduce a Bayesian layer on top of this model by treating the regression parameters $\mathbf{w}_z$ and $\mathbf{w}_x$ probabilistically to protect against overfitting, as shown in Figure 3.3. To do this, we introduce so-called "precision" variables α_m over each regression parameter w_{zm}. The same α_m is also placed over each w_{xm}, leading to a coupled regularization of w_{zm} and w_{xm}. As a result, the regression

parameters are now distributed as $w_{zm} \sim \text{Normal}(0, 1/\alpha_m)$ and $w_{xm} \sim \text{Normal}(0, 1/\alpha_m)$, where α_m takes on a Gamma distribution with parameters a_{α_m} and b_{α_m}, shown below:

$$\begin{aligned} \mathbf{w}_z|\boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Normal}\left(w_{zm}; 0, 1/\alpha_m\right) \\ \mathbf{w}_x|\boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Normal}\left(w_{xm}; 0, 1/\alpha_m\right) \\ \boldsymbol{\alpha} &\sim \prod_{m=1}^{d} \text{Gamma}\left(\alpha_m; a_{\alpha_m,0}, b_{\alpha_m,0}\right) \end{aligned} \tag{3.8}$$

where $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ are initial hyperparameter values for the distribution of α_m. The rationale of this Bayesian modeling technique is as follows. The key quantity that determines the relevance of a regression input is the parameter α_m. *A priori*, we assume that every w_{zm} has a mean-zero distribution with broad variance $1/\alpha_m$. We also assume that the precision α_m has an initial value 1 with large variance by setting both the initial values $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ to 10^{-6}. If the posterior value of α_m turns out to be very large after all model parameters are estimated, then the corresponding posterior distribution of w_{zm} must be sharply peaked at zero. Thus, this gives strong evidence that $w_{zm} = 0$ and that the input t_m contributes no information to the regression model. If an input t_m contributes no information to the output, then it is also irrelevant how much it contributes to x_{im}. That is to say, the corresponding inputs x_m could be treated as pure noise. Coupling both w_{zm} and w_{xm} with the same precision variable α_m accomplishes exactly this effect. In this way, the Bayesian approach automatically detects irrelevant input dimensions and regularizes against ill-conditioned data sets.

Even with the Bayesian layer added, the entire regression problem can be treated as an EM-like learning problem (Ghahramani & Beal 2000). Our goal is to maximize

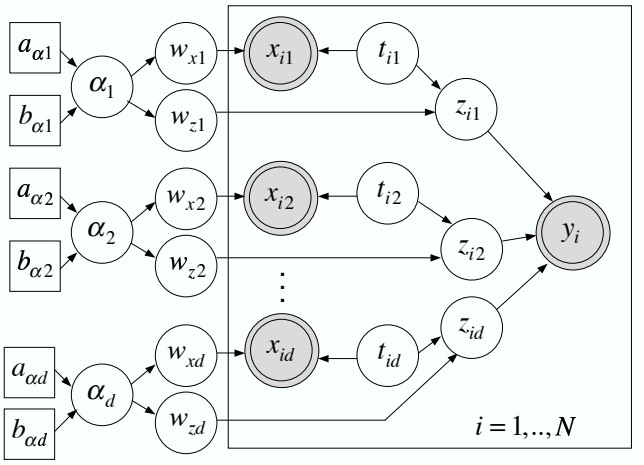


Figure 3.3: Graphical model for Bayesian version of joint factor analysis for noisy linear regression. Random variables are in circular nodes, observed random variables are in shaded double circles, and point estimated parameters are in square nodes. Note that square nodes for the noise variances ψ_y and ψ_{zm} have been omitted from the model for graphical clarity.

the log likelihood $\log p(\mathbf{y}|\mathbf{X})$, i.e., the 'incomplete" log likelihood, as all hidden probabilistic variables are marginalized out. Due to analytical problems, we do not have access to this incomplete log likelihood, but rather only to a lower bound of it. This lower bound is based on an expected value of the so-called "complete" data likelihood, $\langle \log p(\mathbf{y}, \mathbf{Z}, \mathbf{T}, \mathbf{w}_z, \mathbf{w}_x, \boldsymbol{\alpha}|\mathbf{X}) \rangle$, formulated over all variables of the learning problem, where:

$$\begin{aligned}
&\log p(\mathbf{y}, \mathbf{Z}, \mathbf{T}, \mathbf{w}_z, \mathbf{w}_x, \boldsymbol{\alpha}|\mathbf{X}) \\
&= \sum_{i=1}^{N} \log p(y_i|\mathbf{z}_i) + \sum_{i=1}^{N}\sum_{m=1}^{d} \log p(z_{im}|w_{zm}, t_{im}) + \sum_{i=1}^{N}\sum_{m=1}^{d} \log p(x_{im}|w_{xm}, t_{im}) \\
&\quad + \sum_{i=1}^{N}\sum_{m=1}^{d} \log p(t_{im}) + \sum_{m=1}^{d} \log \{p(w_{zm}|\alpha_m)p(\alpha_m)\} \\
&\quad + \sum_{m=1}^{d} \log \{p(w_{xm}|\alpha_m)p(\alpha_m)\} + \text{const}_{\mathbf{y},\mathbf{Z},\mathbf{T},\mathbf{w}_z,\mathbf{w}_x,\boldsymbol{\alpha}}
\end{aligned} \tag{3.9}$$

and where $\mathbf{Z} \in \Re^{N \times d}$ with the vector $\mathbf{z}_i$ in its rows and $\mathbf{T} \in \Re^{N \times d}$ with the vector $\mathbf{t}_i$ in its rows. The expectation of this complete data likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q(\boldsymbol{\alpha}, \mathbf{w}_z, \mathbf{w}_x, \mathbf{Z}, \mathbf{T})$. Unfortunately, this is an analytically intractable expression. Instead, a lower bound can be formulated using a technique from variational calculus where we make a factorial approximation of the true posterior in terms of: $Q(\boldsymbol{\alpha}, \mathbf{w}_z, \mathbf{w}_x, \mathbf{Z}, \mathbf{T}) = Q(\boldsymbol{\alpha})Q(\mathbf{w}_z)Q(\mathbf{w}_x)Q(\mathbf{Z}, \mathbf{T})$. We now have a mechanism that infers the significance of each dimension's contribution to the observed output $\mathbf{y}$ and observed inputs $\mathbf{X}$.

We can derive the EM update equations using standard manipulations of Normal and Gamma distributions (please refer to Appendix C.2 for derivations), reaching the following:

E-step :

$$\sigma^2_{w_{zm}} = \frac{1}{\frac{1}{\psi_{zm}} \sum_{i=1}^{N} \langle t^2_{im} \rangle + \langle \alpha_m \rangle} \tag{3.10}$$

$$\langle w_{zm} \rangle = \frac{\sigma^2_{w_{zm}}}{\psi_{zm}} \sum_{i=1}^{N} \langle z_{im} t_{im} \rangle \tag{3.11}$$

$$\sigma^2_{w_{xm}} = \frac{1}{\frac{1}{\psi_{xm}} \sum_{i=1}^{N} \langle t^2_{im} \rangle + \langle \alpha_m \rangle} \tag{3.12}$$

$$\langle w_{xm} \rangle = \frac{\sigma^2_{w_{xm}}}{\psi_{xm}} \sum_{i=1}^{N} x_{im} \langle t_{im} \rangle \tag{3.13}$$

$$\hat{a}_{\alpha_m} = a_{\alpha_m,0} + 1 \tag{3.14}$$

$$\hat{b}_{\alpha_m} = b_{\alpha_m,0} + \frac{\langle w^2_{zm} \rangle + \langle w^2_{xm} \rangle}{2} \tag{3.15}$$

M-step :

$$\psi_y = \frac{1}{N}\sum_{i=1}^{N}\left(y_i^2 - 2\mathbf{1}y_i\langle \mathbf{z}_i\rangle + \mathbf{1}^T\langle \mathbf{z}_i\mathbf{z}_i^T\rangle\mathbf{1}\right) \tag{3.16}$$

$$\psi_{zm} = \frac{1}{N}\sum_{i=1}^{N}\left(\langle z_{im}^2\rangle - 2\langle w_{zm}\rangle\langle z_{im}t_{im}\rangle + \langle w_{zm}^2\rangle\langle t_{im}^2\rangle\right) \tag{3.17}$$

$$\psi_{xm} = \frac{1}{N}\sum_{i=1}^{N}\left(x_{im}^2 - 2\langle w_{xm}\rangle\langle t_{im}\rangle x_{im} + \langle w_{xm}^2\rangle\langle t_{im}^2\rangle\right) \tag{3.18}$$

where the covariance matrix, $\mathbf{\Sigma}$, of the joint posterior distribution of $\mathbf{Z}$ and $\mathbf{T}$ is $\begin{bmatrix} \mathbf{\Sigma}_{zz} & \mathbf{\Sigma}_{zt} \\ \mathbf{\Sigma}_{tz} & \mathbf{\Sigma}_{tt} \end{bmatrix}$, with:

$$\mathbf{\Sigma}_{zz} = \mathbf{M} - \frac{\mathbf{M}\mathbf{1}\mathbf{1}^T\mathbf{M}}{\psi_y + \mathbf{1}^T\mathbf{M}\mathbf{1}} \tag{3.19}$$

$$\mathbf{\Sigma}_{tt} = \mathbf{K}^{-1} + \mathbf{K}^{-1}\langle \mathbf{W}_z\rangle^T\mathbf{\Psi}_z^{-1}\mathbf{\Sigma}_{zz}\mathbf{\Psi}_z^{-1}\langle \mathbf{W}_z\rangle\mathbf{K}^{-1} \tag{3.20}$$

$$\mathbf{\Sigma}_{zt} = -\mathbf{\Sigma}_{zz}\langle \mathbf{W}_z\rangle\mathbf{\Psi}_z^{-1}\mathbf{K}^{-1} \tag{3.21}$$

$$\mathbf{\Sigma}_{tz} = \mathbf{\Sigma}_{zt}^T \tag{3.22}$$

$$\mathbf{K} = \mathbf{I} + \langle \mathbf{W}_x^T\mathbf{W}_x\rangle\mathbf{\Psi}_x^{-1} + \langle \mathbf{W}_z^T\mathbf{W}_z\rangle\mathbf{\Psi}_z^{-1} \tag{3.23}$$

$$\mathbf{M} = \mathbf{\Psi}_z + \langle \mathbf{W}_z\rangle\left(\mathbf{I} + \langle \mathbf{W}_x^T\mathbf{W}_x\rangle\mathbf{\Psi}_x^{-1} + (\mathbf{\Sigma}_{\mathbf{W}_z})_{mm}\mathbf{\Psi}_z^{-1}\right)^{-1}\langle \mathbf{W}_z\rangle^T \tag{3.24}$$

and where $\langle \mathbf{W}_x\rangle$ is a diagonal d by d matrix with $\langle \mathbf{w}_x\rangle$ along its diagonal. Similarly, $\langle \mathbf{W}_z\rangle$, $\mathbf{\Psi}_x$, $\mathbf{\Psi}_z$ are d by d diagonal matrices with diagonal vectors of $\langle \mathbf{w}_z\rangle, \psi_x$ and ψ_z, respectively. The E-step updates for $\mathbf{Z}$ and $\mathbf{T}$ are then:

$$\langle \mathbf{z}_i\rangle = \frac{y_i}{\psi_y}\mathbf{1}^T\mathbf{\Sigma}_{zz} + x_i\langle \mathbf{W}_x\rangle^T\mathbf{\Psi}_x^{-1}\mathbf{\Sigma}_{tz} \tag{3.25}$$

$$\langle \mathbf{t}_i\rangle = \frac{y_i}{\psi_y}\mathbf{1}^T\mathbf{\Sigma}_{zz}\langle \mathbf{W}_z\rangle\mathbf{\Psi}_z^{-1}\mathbf{K}^{-1} + x_i\langle \mathbf{W}_x\rangle^T\mathbf{\Psi}_x^{-1}\mathbf{\Sigma}_{tt} \tag{3.26}$$

$$\sigma_z^2 = \text{diag}(\mathbf{\Sigma}_{zz}) \tag{3.27}$$

$$\sigma_t^2 = \text{diag}(\mathbf{\Sigma}_{tt}) \tag{3.28}$$

$$\text{cov}(\mathbf{z}, \mathbf{t}) = \text{diag}(\mathbf{\Sigma}_{zt}) \tag{3.29}$$

The final regression solution regularizes over the number of retained inputs in the regression vector, performing a functionality similar to ARD (Neal 1994). It is important to notice that *the resulting generalized EM updates still have a computational complexity of $O(d)$ for each EM iteration*—a level of efficiency that has not been accomplished with previous Joint Factor Analysis regression models, especially with one containing a full Bayesian treatment of JFA regression. The result is an efficient Bayesian algorithm that is robust to high dimensional ill-conditioned noisy data.

Initialization of Parameters: Initialization of parameters is crucial when using the EM algorithm and will affect the quality of the final converged solution. The observed $\{\mathbf{x}_i, y_i\}$ data is assumed to be preprocessed to have a mean of 0 and a variance of 1. Included below is a description of how each parameter is initialized. Note that this set of initializations does not need to be modified from data set to data set.

- w_x: These weights are all initialized to 1, with a little bit of randomness added to them (i.e. $1 + randn(1, d) * 0.1$, where d is the number of input dimensions).

- w_z: These weights are initialized to the correlation vector found using Partial Least Squares ($x^T y$) and normalized such that the length is 1.

- ψ_x: The variance of $\mathbf{x}$ is set to the observed variance of x (that is, 1, since $\mathbf{x}$ has been preprocessed already).

- ψ_z: The variance of the hidden variable $\mathbf{z}$ is set to 1 with a little bit of randomness added $(1 + randn(1, d) * 0.1)$.

- ψ_y: ψ_y is initialized to 1.

- $\{a_{\alpha_m,0}, b_{\alpha_m,0}\}$: Both $a_{\alpha_m,0}$ and $b_{\alpha_m,0}$ are initialized to 10^{-8} so that the prior over α_m is flat and uninformative. That is to say, the precision α_m has an initial mean of 1, with a very large variance.

For details on how to check for convergence of the EM algorithm, please refer to Appendix C.3.

3.2.3 Inference of the Regression Solution

Estimating the rather complex probabilistic Bayesian model for Joint Factor Analysis regression gives us the distributions and mean values for all hidden variables. However, one additional step is required to infer the final regression parameters, which, in our application, are the RBD parameters. For this purpose, we consider the predictive distribution $p(y^q|\mathbf{x}^q)$ for a new noisy test input $\mathbf{x}^q$ and its unknown output y^q. We can calculate $\langle y^q|\mathbf{x}^q\rangle$, the mean of the distribution associated with $p(y^q|\mathbf{x}^q)$, by conditioning y^q on $\mathbf{x}^q$ and marginalizing out all hidden variables. Since an analytical solution of the resulting integral is only possible for the probabilistic Joint Factor Analysis regression model in Figure 3.2 and not for the full Bayesian treatment, we restrict our computations to this simpler probabilistic model, and assume that $\mathbf{W}_x$ and $\mathbf{W}_z$ are replaced by their

point estimates $\langle \mathbf{W}_x \rangle$ and $\langle \mathbf{W}_z \rangle$, such that our results will hold in approximation for the Bayesian model.

Thus, the predictive distribution is:

$$p(y^q|\mathbf{x}^q, \mathbf{X}, \mathbf{Y}) = \int\int p(y^q, \mathbf{Z}, \mathbf{T}|\mathbf{x}^q, \mathbf{X}, \mathbf{Y}) d\mathbf{Z} d\mathbf{T} \tag{3.30}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are noisy input and noisy output data used for training. From solving this integral, can infer the value of the regression estimate $\hat{\mathbf{b}}$ since $\langle y^q|\mathbf{x}^q \rangle = \hat{\mathbf{b}}^T \mathbf{x}^q$. The resulting regression estimate, given noisy inputs $\mathbf{x}^q$ and noisy outputs y^q, is $\hat{\mathbf{b}}_{\text{noise}}$:

$$\hat{\mathbf{b}}_{\text{noise}} = \frac{\psi_y \mathbf{1}^T \mathbf{B}^{-1}}{\psi_y - \mathbf{1}^T \mathbf{B}^{-1} \mathbf{1}} \mathbf{\Psi}_z^{-1} \langle \mathbf{W}_z \rangle \mathbf{A}_{\text{noise}}^{-1} \langle \mathbf{W}_x \rangle^T \mathbf{\Psi}_x^{-1} \tag{3.31}$$

where $\mathbf{\Psi}_x$ is a diagonal matrix with the vector ψ_x on its diagonal ($\langle \mathbf{W}_x \rangle$, $\langle \mathbf{W}_z \rangle$, $\mathbf{\Psi}_z$ are similarly defined diagonal matrices with vectors of $\langle \mathbf{w}_x \rangle$, $\langle \mathbf{w}_z \rangle$ and ψ_z on their diagonals, respectively) and where:

$$\mathbf{A}_{\text{noise}} = \mathbf{I} + \left\langle \mathbf{W}_x^T \mathbf{W}_x \right\rangle \mathbf{\Psi}_x^{-1} + \left\langle \mathbf{W}_z^T \mathbf{W}_z \right\rangle \mathbf{\Psi}_z^{-1} \tag{3.32}$$

$$\mathbf{B} = \left(\frac{\mathbf{1}\mathbf{1}^T}{\psi_y} + \mathbf{\Psi}_z^{-1} - \mathbf{\Psi}_z^{-1} \langle \mathbf{W}_z \rangle^T \mathbf{A}^{-1} \langle \mathbf{W}_z \rangle \mathbf{\Psi}_z^{-1} \right) \tag{3.33}$$

If we compare $\hat{\mathbf{b}}_{\text{noise}}$ in Eq. (3.31) to $\hat{\mathbf{b}}_{\text{JFA}}$, the regression estimate derived for Joint Factor Analysis regression (which can be arrived at also by conditioning y on $\mathbf{x}$ and marginalizing the latent variables) is:

$$\hat{\mathbf{b}}_{\text{JFA}} = \mathbf{W}_z \mathbf{A}_{\text{JFA}}^{-1} \mathbf{W}_x^T \Psi_x^{-1} \tag{3.34}$$

$$\mathbf{A}_{\text{JFA}} = \mathbf{I} + \mathbf{W}_x^T \mathbf{W}_x \Psi_x^{-1} \mathbf{W}_x$$

we can see that $\hat{\mathbf{b}}_{\text{noise}}$ contains an additional term $\left\langle \mathbf{W}_z^T \mathbf{W}_z \right\rangle \mathbf{\Psi}_z^{-1}$ in its $\mathbf{A}$ expression, due to the introduction of hidden variables $\mathbf{z}$. $\hat{\mathbf{b}}_{\text{noise}}$ is scaled by an additional variance-related term because of this issue as well.

It is important to note that the regression vector $\hat{\mathbf{b}}_{\text{noise}}$ given by Eq. (3.31) is for optimal prediction from *noisy* input data. However, for system identification in RBD, we are interested in obtaining the true regression vector, which is the regression vector that predicts output from *noiseless* inputs. Thus, the result in Eq. (3.31) is not quite suitable and what we want to calculate is the mean of $p(y^q|\mathbf{t}^q)$, where $\mathbf{t}^q$ are noiseless inputs.

To address this issue, we can take the limit of $\hat{\mathbf{b}}_{\text{noise}}$ by letting $\psi_x \to 0$ and interpret the resulting expression to be the true regression vector for noiseless inputs (as $\psi_x \to 0$, the amount of input noise approaches 0). The resulting regression vector estimate $\hat{\mathbf{b}}_{\text{true}}$ becomes:

$$\hat{\mathbf{b}}_{\text{true}} = \frac{\psi_y \mathbf{1}^T \mathbf{C}^{-1}}{\psi_y - \mathbf{1}^T \mathbf{C}^{-1} \mathbf{1}} \mathbf{\Psi}_z^{-1} \left\langle \mathbf{W}_z \right\rangle^T \left\langle \mathbf{W}_x \right\rangle^{-1} \tag{3.35}$$

where $\mathbf{C} = \left(\frac{\mathbf{1}\mathbf{1}^T}{\psi_y} + \mathbf{\Psi}_z^{-1} \right)$, and this is the desired regression vector estimate for noiseless data that we use in our evaluations.

3.3 Post-processing for Physically Consistent Rigid Body Parameters

Before our evaluations, we need to return for a moment to the specifics of our intended application domain of RBD parameter estimation. Given a Bayesian estimate of the RBD parameters, we would like to ensure that the inferred regression vector satisfies the constraints given by positive definite inertia matrices and the parallel axis theorem.

In our RBD estimation problem, there are 11 RBD parameters for each DOF, which we arrange in an 11-dimensional vector $\boldsymbol{\theta}$ consisting of the following parameters: mass, three center of mass coefficients multiplied by the mass and six inertial parameters. This choice of parameterization is the only one that is identifiable using linear regression (An et al. 1988). Additionally, we include viscous friction as the 11th parameter.

In order to enforce the aforementioned physical constraints, we introduce a 11-dimensional virtual parameter vector $\hat{\boldsymbol{\theta}}$ that we assume is used in a nonlinear transformation to generate $\boldsymbol{\theta}$, e.g., $\boldsymbol{\theta} = f(\hat{\boldsymbol{\theta}})$. This nonlinear transformation between virtual parameters $\hat{\boldsymbol{\theta}}$ and actual parameters $\boldsymbol{\theta}$ is shown below for one DOF:

$$
\begin{aligned}
\boldsymbol{\theta}_1 &= \hat{\boldsymbol{\theta}}_1^2 & \boldsymbol{\theta}_2 &= \hat{\boldsymbol{\theta}}_2\hat{\boldsymbol{\theta}}_1^2 \\
\boldsymbol{\theta}_3 &= \hat{\boldsymbol{\theta}}_3\hat{\boldsymbol{\theta}}_1^2 & \boldsymbol{\theta}_4 &= \hat{\boldsymbol{\theta}}_4\hat{\boldsymbol{\theta}}_1^2 \\
\boldsymbol{\theta}_5 &= \hat{\boldsymbol{\theta}}_5^2 + \left(\hat{\boldsymbol{\theta}}_4^2 + \hat{\boldsymbol{\theta}}_3^2\right)\hat{\boldsymbol{\theta}}_1^2 & \boldsymbol{\theta}_6 &= \hat{\boldsymbol{\theta}}_5\hat{\boldsymbol{\theta}}_6 - \hat{\boldsymbol{\theta}}_2\hat{\boldsymbol{\theta}}_3\hat{\boldsymbol{\theta}}_1^2 \\
\boldsymbol{\theta}_7 &= \hat{\boldsymbol{\theta}}_5\hat{\boldsymbol{\theta}}_7 - \hat{\boldsymbol{\theta}}_2\hat{\boldsymbol{\theta}}_4\hat{\boldsymbol{\theta}}_1^2 & \boldsymbol{\theta}_8 &= \hat{\boldsymbol{\theta}}_6^2 + \hat{\boldsymbol{\theta}}_8^2 + \left(\hat{\boldsymbol{\theta}}_2^2 + \hat{\boldsymbol{\theta}}_4^2\right)\hat{\boldsymbol{\theta}}_1^2 \\
\boldsymbol{\theta}_9 &= \hat{\boldsymbol{\theta}}_6\hat{\boldsymbol{\theta}}_7 + \hat{\boldsymbol{\theta}}_8\hat{\boldsymbol{\theta}}_9 - \hat{\boldsymbol{\theta}}_3\hat{\boldsymbol{\theta}}_4\hat{\boldsymbol{\theta}}_1^2 & \boldsymbol{\theta}_{10} &= \hat{\boldsymbol{\theta}}_7^2 + \hat{\boldsymbol{\theta}}_9^2 + \hat{\boldsymbol{\theta}}_{10}^2 + \left(\hat{\boldsymbol{\theta}}_2^2 + \hat{\boldsymbol{\theta}}_3^2\right)\hat{\boldsymbol{\theta}}_1^2 \\
\boldsymbol{\theta}_{11} &= \hat{\boldsymbol{\theta}}_{11}^2
\end{aligned}
\tag{3.36}
$$

In essence, the virtual parameters $\hat{\boldsymbol{\theta}}$ correspond to the square root of the mass, the true center-of-mass coordinates (i.e., not multiplied by the mass), a Cholesky decomposition of the DOF's inertial matrix at the center of gravity to ensure positive definiteness of the inertial matrix, and the square root of the viscous friction coefficient. The functions in Eq. (3.36) encode the parallel axis theorem and some additional constraints, ensuring that the mass and viscous friction coefficients remain strictly positive. Given the above formulation, any arbitrary set of virtual parameters gives rise to a physically consistent set of actual parameters for the RBD problem. For a robotic system with s DOFs, Eq. (3.36) is repeated for each DOF. Since there are 11 features for each DOF, the result is a $11s$-dimensional regression vector $\boldsymbol{\theta}$, where $\boldsymbol{\theta}_m = f_m(\hat{\boldsymbol{\theta}})$ (for $m = 1..d$ where $d = 11s$),

There are at least two possible ways to enforce the physical constraints of RBD parameters in our Bayesian estimation algorithm. The first (ideal) approach involves reformulating our algorithm using the virtual parameters $\hat{\boldsymbol{\theta}}$ described previously instead of the actual parameters $\boldsymbol{\theta}$. Unfortunately, this method will lead to an analytically intractable set of update equations due to the nonlinear relationship between virtual and actual parameters. In the second approach, we can consider a post-processing step, where the unconstrained parameters are appropriately projected onto the constrained parameters. For this purpose, we assume that we would like to find the optimal virtual parameters in a least squares sense, i.e., by minimizing the cost function:

$$J = \left\langle \frac{1}{2} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta} \right)^T \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta} \right) \right\rangle \tag{3.37}$$

where $\mathbf{X}$ and $\mathbf{y}$ are input and output data, and we have the constraints of $\theta_m = f_m(\hat{\boldsymbol{\theta}})$. For the moment, we will ignore issues of noise in input data and ill-conditioned data sets. Let us assume that some arbitrary estimation algorithm generated an estimate for the unconstrained parameters as $\boldsymbol{\theta}_{uc}$. Thus, the constrained parameters can be written as $\boldsymbol{\theta} = \boldsymbol{\theta}_{uc} + \Delta\boldsymbol{\theta}$, where $\Delta\boldsymbol{\theta}$ denotes the difference between constrained and unconstrained parameters. Substituting this into Eq. (3.37) results in:

$$\begin{aligned} J &= \left\langle \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \right\rangle \\ &= \frac{1}{2} \left\langle (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc}) \right\rangle - \left\langle (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc})^T \mathbf{X}\Delta\boldsymbol{\theta} \right\rangle + \frac{1}{2} \left\langle \Delta\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X}\Delta\boldsymbol{\theta} \right\rangle \end{aligned} \tag{3.38}$$

Minimizing this cost function with respect to the virtual parameters only requires consideration of the second and third terms of Eq. (3.38) since the first term does not depend on the virtual parameters.

Now, let us consider algorithms to generate $\boldsymbol{\theta}_{uc}$. Among the most straightforward algorithms is OLS, which is equivalent to reformulating Eq. (3.37) in terms of $\boldsymbol{\theta}_{uc}$:

$$J_{uc} = \left\langle \frac{1}{2} (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc}) \right\rangle, \tag{3.39}$$

taking the derivative $\frac{\partial J_{uc}}{\partial \boldsymbol{\theta}_{uc}}$ and setting it to zero:

$$\frac{\partial J}{\partial \boldsymbol{\theta}_{uc}} = -(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}_{uc})^T \mathbf{X} = 0 \tag{3.40}$$

If we insert this result into Eq. (3.38), we see that the second term of this cost function equals zero, leaving only the third term to be considered in order to obtain the optimal

virtual parameters. Thus, we can conclude that for optimal projection of the unconstrained parameters onto the constrained parameters, all we need to do is to minimize the difference between unconstrained and constrained parameters under the metric $\mathbf{X}^T\mathbf{X}$.

We can consider other algorithms (other than OLS) to generate $\boldsymbol{\theta}_{uc}$. For instance, SVD regression (Belsley et al. 1980) performs OLS in a subspace of the original input dimensionality of the regression problem. Thus, the cost functions in Eqs. (3.39) and (3.38) would be formulated only over the input dimensions that were identified to be relevant to the regression problem. Hence, the results regarding the minimization of the difference between unconstrained and constrained parameters hold as well.

More interestingly, if we use our Bayesian estimation method to generate $\boldsymbol{\theta}_{uc}$, the result will be similar to SVD regression in that some of the input dimensions will be eliminated. Additionally, the algorithm also estimates the noise in the inputs and returns a regression vector that can be applied to noiseless query points. If we re-express the noisy inputs $\mathbf{X}$ as $\mathbf{X}_t + \Gamma$, where $\mathbf{X}_t$ are noiseless inputs and Γ is the input noise, then we can re-write the third term of Eq. (3.38) in terms of de-noised quantities:

$$\frac{1}{2}\Delta\boldsymbol{\theta}^T \left(\mathbf{X}_t^T\mathbf{X}_t\right)\Delta\boldsymbol{\theta} \tag{3.41}$$

The second term of Eq. (3.38) does not yield exactly zero as in an OLS regression, but, empirically, it is very close to zero, such that only the term in Eq. (3.41) matters in the actual optimization problem.

In summary, we can see that in order to minimize the least squared error in Eq. (3.37) with respect to the physically constrained parameters of RBD, we can follow an approximate two-step procedure. First, we apply our Bayesian algorithm (or any other algorithm, for that matter) to come up with an optimal unconstrained parameter estimate $\boldsymbol{\theta}_{uc}$. Then, we find the virtual parameter estimates $\hat{\boldsymbol{\theta}}$ (and the corresponding physically consistent parameter estimates $\boldsymbol{\theta}$) such that the error between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{uc}$ is minimized in the sense of Eq. (3.41). If the noiseless inputs are not estimated explicitly, the term $\mathbf{X}_t$ is replaced by the noisy inputs $\mathbf{X}$. The optimization of Eq. (3.41) is easily achieved numerically as it is a simple convex function with a unique global minimum. If $\boldsymbol{\theta}_{uc}$ is estimated by OLS or SVD regression, the results for the constrained parameters are optimal. If $\boldsymbol{\theta}_{uc}$ is estimated by our Bayesian or any other nonlinear method, the results for the constrained parameters are approximately optimal. Empirically, we found that the above proposed procedure always achieves satisfying results.

3.4 Evaluation

We evaluated our algorithm on both synthetic data and robotic data for the task of system identification. The goal of these evaluations was to determine how well our Bayesian de-noising algorithm performs compared to other standard techniques for parameter estimation in the presence of noisy input and noisy output data.

First, we start by evaluating our algorithm on a synthetic dataset in order to illustrate its effectiveness at de-noising input and output data. Then, we apply the algorithms on a 7 DOF robotic oculomotor vision head and a 10 DOF robotic anthromorphic arm, both

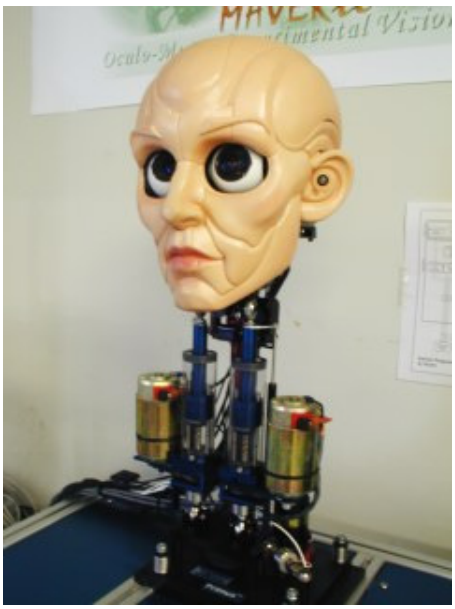


Figure 3.4: Robotic oculomotor vision head by Sarcos (Cambridge, MA).

shown in Figures 3.4 and 3.4, respectively, for the task of parameter estimation in rigid body dynamics.

3.4.1 Synthetic Data

We synthesized random input training data consisting of 10 relevant dimensions and 90 irrelevant and redundant dimensions. The first 10 input dimensions were drawn from a multi-dimensional Gaussian distribution with a random covariance matrix. The output data was generated using an ordered regression vector $\mathbf{b}_{\text{true}} = [1, 2, ..., 10]^T$. Output noise was added with a signal-to-noise ratio (SNR) of 5. Then, we added Gaussian noise with varying SNRs (a SNR of 2 for strongly noisy input data and a SNR of 5 for less noisy input data) to the relevant 10 input dimensions. A varying number of redundant data vectors was added to the input data, and these were generated from random convex

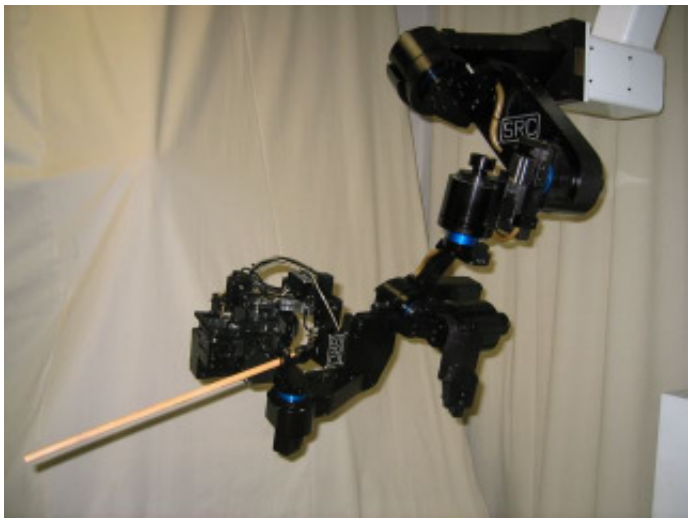
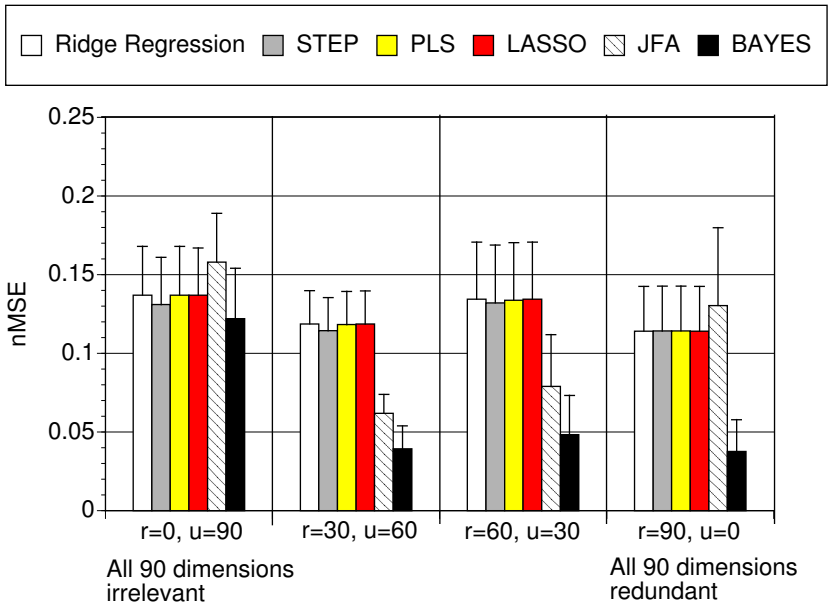


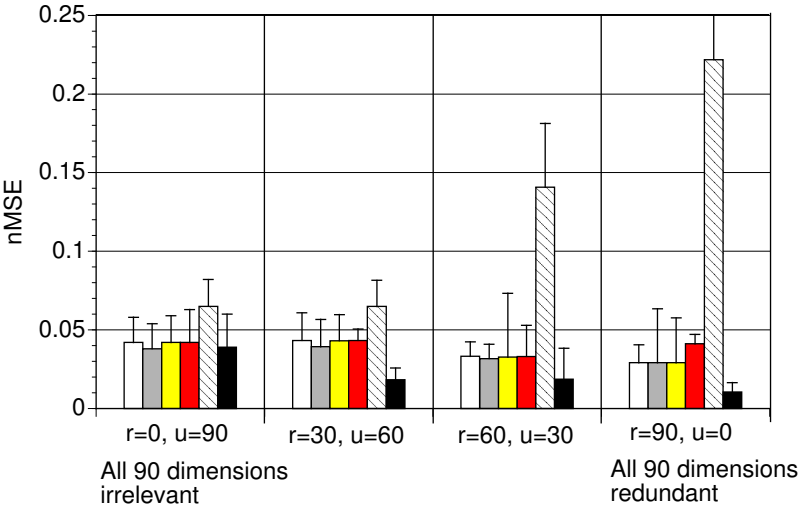
Figure 3.5: Robotic anthropomorphic arm by Sarcos (Cambridge, MA).

combinations of the 10 noisy relevant data vectors. Finally, we added irrelevant data columns, drawn from a Normal$(0, 1)$ distribution, until a total of 100 input dimensions were attained. The result was an input training dataset that contained irrelevant and redundant dimensions. Test data was created using the same method outlined above, except that input and output data were both noiseless.

We compared our Bayesian de-noising algorithm with the following methods: i) OLS regression; ii) stepwise regression (Draper & Smith 1981), which tends to be inconsistent in the presence of collinear inputs (Derksen & Keselman 1992); iii) PLS regression (Wold 1975), a slightly heuristic but empirically successful regression method for high dimensional data; iv) LASSO regression (Tibshirani 1996), which gives sparse solutions by shrinking certain coefficients to zero under the control of a regularization parameter; v) our probabilistic treatment of Joint Factor Analysis regression in Figure 3.2; and vi)



(a) Average prediction error for training data with a SNR of 2



(b) Average prediction error for training data with a SNR of 5

Figure 3.6: Average normalized mean squared prediction errors (nMSE) on noiseless test data for a 100 dimensional dataset with 10 relevant input dimensions and various combinations of redundant input dimensions v and irrelevant input dimensions u, averaged over 10 trials. Output data has SNR = 5. Algorithms evaluated include OLS, stepwise regression (STEP), PLS regression (PLS), LASSO regression (LASSO), Joint Factor Analysis regression (JFA) and our Bayesian de-noising algorithm (BAYES).

our Bayesian de-noising algorithm shown in Figure 3.3. In this synthetic evaluation, there was no need to constrain parameters according to some physical consistency rules.

The Bayesian de-noising algorithm had an improvement of 10 to 300% compared to other algorithms, as the black bars in Figures 3.6(a) and 3.6(b) illustrate. One interesting observation is that for the case where the 90 input dimensions are all irrelevant, the Bayesian de-noising algorithm did not give a significant reduction in error as in the other three scenarios. This result can be explained by the fact that the other algorithms suffer primarily from redundant inputs, but not so much from irrelevant inputs, which does not cause numerical problems. The true power of our Bayesian algorithm lies in its ability to identify relevant dimensions in the presence of redundant and irrelevant data.

3.4.2 Robotic Oculomotor Vision Head

Table 3.1: Root mean squared errors for position (in radians), velocity (radians/sec) and feedback command (in Newton-meters) for the robotic oculomotor vision head.

Algorithm	Position (rad)	Velocity (rad/s)	Feedback (Nm)
Ridge Regression	0.0291	0.2465	0.3969
Bayesian De-noising	0.0243	0.2189	0.3292
LASSO Regression	0.0308	0.2517	0.4274
Stepwise Regression	FAILURE	FAILURE	FAILURE

The Sarcos robotic oculomotor vision head, shown in Figure 3.4, has 7 DOFs, giving 77 features in total (there are 11 features per DOF). The kinematic structure of robotic systems always creates non-identifiable parameters and thus, redundancies (An et al. 1988). We implemented a computed torque control law on the robot, using estimated parameters from each technique. Table 3.1 shows the root mean squared errors averaged over all DOFs. The Bayesian parameter estimation approach performed around 10 to

20% better than ridge regression with gradient descent, well LASSO regression performed worse. Stepwise regression produced RBD parameters that were physically impossible to run on the robotic head. This can be attributed to stepwise regression's failure to identify the relevant features in the data set.

3.4.3 Robotic Anthropomorphic Arm

We also evaluated the parameter estimation algorithms on a 10 DOF Sarcos robotic anthropomorphic arm, shown in Figure 3.4, and evaluations were done in a similar way as for the robot head. We collected about a million data points from the arm and downsampled the data to a more manageable size of 500,000. Table 3.2 shows the results averaged over all 10 DOFs. The Bayesian parameter estimation approach performed around 5 to 17% better than the other techniques. LASSO regression failed, due to over-aggressive clipping of relevant dimensions, and stepwise regression produced RBD parameters that were impossible to run on the robotic arm.

Table 3.2: Root mean squared errors for position (in radians), velocity (radians/sec) and feedback command (in Newton-meters) for the robotic anthropomorphic arm.

Algorithm	Position (rad)	Velocity (rad/s)	Feedback (Nm)
Ridge Regression	0.0210	0.1119	0.5839
Bayesian De-noising	0.0201	0.0930	0.5297
LASSO regression	FAILURE	FAILURE	FAILURE
Stepwise regression	FAILURE	FAILURE	FAILURE

3.5 Discussion

This chapter addresses the problem of learning for system identification, as, for example, in a scenario where we have observed a system through empirical data and would like to uncover its true parameters. Learning for system identification differs from learning for prediction. Learning for prediction is the more common problem setting in many machine learning techniques for regression. Good prediction is often possible without modeling all components of the generative system. For instance, linear regression with noise in both the input and output data achieves surprisingly good prediction results on test data that has the same noise properties as the training data, despite the well-known fact that linear regression is not built to deal with noise in the input data.

The interesting new component of system identification comes from the desire to use the identified model in other ways than in the training scenario. In robotics, a typical example is the use of the system model for prediction with noiseless input data. In this scenario, the training data might have been contaminated by a large amount of input noise. Another typical application is to create an analytical inverse of an identified model as often needed in model-based control. For such applications, the system model needs to be identified as accurately as possible. This is only possible if all parameters of the data generating model (in particular all noise processes) are identified accurately.

We address linear system identification for situations where noise exists in both input and output data—a typical case in most robotic applications where data is derived from noisy sensors. Additionally, we allow for the case of high-dimensional data, where many input dimensions are potentially redundant or irrelevant. To date, no efficient and robust

algorithm has been suggested for such a problem setup. Inspired by factor analysis regression, a classical machine learning technique, we develop a novel full Bayesian treatment of the linear system identification problem. Due to effective Bayesian regularization, this algorithm is robust to high dimensional, ill-conditioned data with noise-contaminated input and output data and remains computationally efficient, i.e., $O(d)$ per iteration of the underlying EM-like algorithm, where d is the number of input dimensions. This algorithm has no parameters that need manual tuning. The algorithm is, however, iterative, but so is probabilistic factor analysis. The iterative nature of the algorithm allows it to be embedded into other more complex, iterative methods and makes it suitable for real-time learning scenarios.

We used this algorithm to estimate parameters in rigid body dynamics—an estimation problem that is linear in the unknown parameters. Since these parameters have physical meaning, it was necessary to enforce physical consistent parameters with a post-processing step. The physical constraints arose from positive definiteness of inertia matrices, positiveness of mass parameters, and the parallel axis theorem. We demonstrated the efficiency of our algorithm by applying it to a synthetic dataset, a 7 DOF robotic vision head and a 10 DOF robotic anthropomorphic arm. Our algorithm successfully identified the system parameters with 10 to 300% higher accuracy than alternative methods on synthetic data for parameter estimation in linear regression. It performed 5 to 25% better on real robot data, proving to be a competitive alternative for parameter estimation on complex high degree-of-freedom robotic systems.

If desired, our Bayesian algorithm can easily be extended to nonlinear system identification in the framework of Locally Weighted Learning (LWL) (Atkeson et al. 1997).

The only modification needed is to change the linear regression problem to a Bayesian weighted linear regression problem (Gelman et al. 2000). Thus, a piecewise linear model identification can be achieved, similar to Schaal & Atkeson (1998) and (Vijayakumar, D'Souza & Schaal 2005). Parameters identified in such a nonparametric way usually lack any physical interpretability, such that our suggested post-processing to enforce physical correctness of the parameters is not applicable. We will leave the problem of full nonlinear system identification for future work.

Chapter 4

Dealing with Outlier-Infested Data

Robotic systems and their control mechanisms rely crucially on the quality of sensory data in order to make robust control decisions. While certain sensors such as potentiometers or optical encoders are inherently easy to assess in their noise characteristics, other sensors such as visual systems, global positioning system (GPS) devices and sonar sensors may provide measurements that are infested by outliers. Thus, robust and reliable outlier removal is necessary in order to include these types of data in control processes. The particular application domain of legged locomotion is especially vulnerable to perceptual data of poor quality, as one undetected outlier can potentially disturb the balance controller to the point that the robot loses stability.

Additionally, for real-time applications, storing data samples may not be an option due to the high frequency of sensory data and insufficient memory resources. In this scenario, sensor data is made available one sample at a time (arriving sequentially over time) and must be discarded once they have been observed.

4.1 Background

An outlier is generally defined as an observation that "lies outside some overall pattern of distribution" (Moore & McCabe 1999). Outliers may arise from sensor noise (producing values that fall outside the valid range of values), temporary sensor failures or unanticipated disturbances in the environment (e.g., a brief change of lighting conditions for a visual sensor). A typical approach of detecting outliers is to characterize what normal observations look like, and then to single out samples that deviate from these normal properties. Existing methods for outlier detection include i) methods that classify a data sample based on a (Mahalanobis) distance from the expected value, ii) approaches that use information-theoretic principles, such as selecting the subset of data points that minimize the prediction error, and iii) techniques that assume that the data was generated by some special generative model.

Outlier classification based on a Mahalanobis distance can work well, but it requires the setting of a threshold that defines whether a point is an outlier or not. This threshold typically is determined using expert domain knowledge or tuned manually beforehand in order to determine its empirically optimal value for the system.

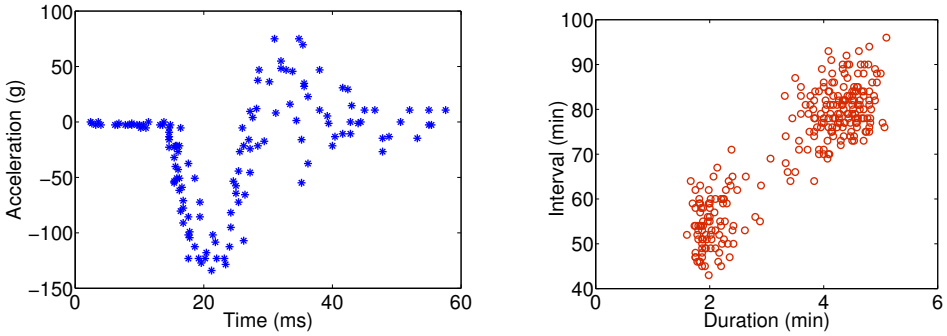
In information-theoretic approaches, outlier detection may be done through active learning (Abe, Zadrozny & Langford 2006), clustering (Breitenbach & Grudic 2005, Ng, Jordan & Weiss 2001) or mixture models (Aitkin & Wilson 1980, Scott 2005). These methods may require sampling, the setting of certain parameters (i.e. the optimal k in k-means clustering (MacQueen 1967)), and may not all lend themselves to a real-time implementation. Another common method is the Random Sample Consensus (RANSAC)

algorithm (Fischler & Bolles 1981). RANSAC tries to find the subset of data samples that produces the lowest error in an iterative fashion. Unfortunately, this may be too computationally intensive for real-time applications and may involve heuristic methods to narrow down the searchable space of subsets.

Mixture models fall into the second and third category, assuming the data was generated by some underlying structure, e.g. a mixture of a Gaussian distribution and a uniform distribution (Fox, Burgard, Dellaert & Thrun 1999, Konolige 2001, Faul & Tipping 2001). The probabilistic assumptions of this approach, however, may be restrictive and may not work as well on data sets where outliers and inliers are not demarked by a large margin. Finally, there also exist outlier-robust versions of standard algorithms such as Independent Component Analysis (ICA) and PCA, e.g., (Hubert, Rousseeuw & Vanden Branden 2005).

The ideal algorithm should detect and remove outliers in real-time—without the need parameter tuning, sampling or model assumptions. In this chapter, we propose a novel Bayesian algorithm that automatically detects outliers in general linear models. We introduce our Bayesian linear regression algorithm, before presenting a modified version that can be implemented in real-time. We evaluate our algorithm on both synthetic and robotic data, demonstrating how it performs at least as well as other standard approaches. In certain cases, it outperforms well-tuned alternative methods. Finally, we extend this idea to the Kalman filter, producing a filter that tracks observations and detects outliers in the observations. We show that this outlier-robust Kalman filter performs as well as other robust Kalman filter algorithms and requires no parameter tuning by the user, offering a competitive alternative to filtering sensor data.

Before going into greater detail, though, it is important to realize that in order to distinguish outliers from inliers, some amount of prior knowledge about the presence of outliers is necessary. As an illustrative example, consider Figures 4.1(a) and 4.1(b), which show the number of motorcycle impacts (Silverman 1985) and time of eruptions for the Old Faithful geyser in Yellowstone National Park (Azzalini & Bowman 1990). It would be tricky to distinguish noise or outliers from the data structure if the source of the data set (i.e., how noisy the data is or how many outliers appear in the data) was not known. This domain knowledge translates naturally into a noise or outlier prior.



(a) Motorcycle impact data set

(b) Old Faithful geyser eruption data set

Figure 4.1: Motorcycle impact data set from (Silverman 1985) and Old Faithful geyser eruption data set (Azzalini & Bowman 1990).

4.2 Linear Regression with Outliers

Given an observed data set $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$ with N data samples, where $\mathbf{x}_i \in \Re^{d\times 1}$, y_i is a scalar and d is the number of input dimensions, we can arrange the input vectors

$\mathbf{x}_i$ in the rows of the matrix $\mathbf{X}$ and set the corresponding scalar outputs y_i to be the coefficients of the vector $\mathbf{y}$. A general model for linear regression is then:

$$y_i = \mathbf{b}^T \mathbf{x}_i + \epsilon_{y_i} \tag{4.1}$$

where $\mathbf{b} \in \Re^{d \times 1}$ and ϵ_{y_i} is additive mean-zero Gaussian noise. The OLS estimate of the regression vector $\mathbf{b}_{\text{OLS}}$ is $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$. However, it is not uncommon for observed data to have outliers, and if outliers are not removed, the regression estimate $\mathbf{b}_{\text{OLS}}$ will be biased.

4.2.1 Bayesian Regression for Automatic Outlier Detection

We can modify Eq. (4.1) so that the observed outputs $\mathbf{y}$ have heteroscedastic (i.e., unequal) variances, introducing a weight w_i for each y_i such that the variance of y_i is weighted with w_i. Gelman et al. (2000) do this, but assume the weights are known beforehand. Using incorrect estimates for the weights may lead to deteriorated performance. As a result, we favor a different approach and treat the weights probabilistically in order to learn them.

Another robust regression algorithm with a Bayesian treatment is that of Faul & Tipping (2001). A Gaussian prior is placed over $\mathbf{b}$, and hyperparameters are introduced in order to have automatic relevance determination on the input data features. However, a mixture model is used to explain outliers (with uniform or Gaussian distribution used to capture them). In contrast, our model makes no assumptions about the underlying

data structure. It is a Bayesian treatment of weighted regression that detects and eliminates outliers automatically (Ting, D'Souza & Schaal 2007). It has the following prior probability distributions on its random variables:

$$\begin{aligned} y_i &\sim \text{Normal}\left(\mathbf{b}^T\mathbf{x}_i, \sigma^2/w_i\right) \\ \mathbf{b} &\sim \text{Normal}\left(\mathbf{b}_0, \mathbf{\Sigma}_{\mathbf{b},0}\right) \\ w_i &\sim \text{Gamma}\left(a_{w_i}, b_{w_i}\right) \end{aligned} \tag{4.2}$$

where $\mathbf{b}_0 \in \Re^{d\times 1}$ is the prior mean of $\mathbf{b}$; $\mathbf{\Sigma}_{\mathbf{b},0}$ is the prior covariance of $\mathbf{b}$ and a d by d diagonal matrix; and σ^2 is the variance of the mean-zero normally distributed output noise.

We can treat the entire regression problem as an EM learning problem. Our goal is to maximize the log likelihood $\log p(\mathbf{y}|\mathbf{X})$. Due to analytical issues, we do not have access to the log likelihood, but instead, only a lower bound of it. The lower bound is based on an expected value of the "complete" data likelihood, $\langle \log p(\mathbf{y}, \mathbf{b}, \mathbf{w}|\mathbf{X})\rangle$, where:

$$\log p(\mathbf{y}, \mathbf{b}, \mathbf{w}|\mathbf{X}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, w_i, \mathbf{b}) + \log p(\mathbf{b}) + \sum_{i=1}^{N} \log p(w_i) \tag{4.3}$$

The expectation of the complete data likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q(\mathbf{b}, \mathbf{w})$. Since this is an analytically intractable expression, a lower bound can be formulated using a technique from variational calculus where we make a factorial approximation of the true posterior as follows: $Q(\mathbf{b}, \mathbf{w}) = Q(\mathbf{b})Q(\mathbf{w})$. While losing a small amount of accuracy, all resulting posterior

distributions over hidden variables become analytically tractable. The final posterior EM-update equations are listed below:

$$\boldsymbol{\Sigma}_{\mathbf{b}} = \left(\boldsymbol{\Sigma}_{\mathbf{b},0}^{-1} + \frac{1}{\sigma^2} \sum_{i=1}^{N} \langle w_i \rangle \, \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \tag{4.4}$$

$$\langle \mathbf{b} \rangle = \boldsymbol{\Sigma}_{\mathbf{b}} \left(\boldsymbol{\Sigma}_{\mathbf{b},0}^{-1} \mathbf{b}_0 + \frac{1}{\sigma^2} \sum_{i=1}^{N} \langle w_i \rangle \, y_i \mathbf{x}_i \right) \tag{4.5}$$

$$\langle w_i \rangle = \frac{a_{w_i,0} + \frac{1}{2}}{b_{w_i,0} + \frac{1}{2\sigma^2} \left(y_i - \langle \mathbf{b} \rangle^T \mathbf{x}_i \right)^2 + \frac{1}{2\sigma^2} \mathbf{x}_i^T \boldsymbol{\Sigma}_{\mathbf{b}} \mathbf{x}_i} \tag{4.6}$$

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} \left[\left(y_i - \langle \mathbf{b} \rangle^T \mathbf{x}_i \right)^2 + \mathbf{x}_i^T \boldsymbol{\Sigma}_{\mathbf{b}} \mathbf{x}_i \right] \tag{4.7}$$

These update equations need to be run iteratively until all parameters and the complete log likelihood converge to steady values.

Examining Eq. (4.6) reveals that if the prediction error in y_i is so large that it dominates over the other denominator terms, then the weight $\langle w_i \rangle$ of that point will be very small. As this prediction error term in the denominator goes to ∞, $\langle w_i \rangle$ approaches 0. As can be seen in both Eqs. (4.4) and (4.5), a data point with an extremely small weight will have a smaller contribution to the calculation of the regression estimate $\langle \mathbf{b} \rangle$. This effect is equivalent to the detection and removal of an outlier if the weight of the data point $(\mathbf{x}_i, y_i)$ is small enough.

Initialization of Priors: A few comments should be made regarding the initialization of the priors used in Eqs. (4.4) to (4.7). First of all, $\boldsymbol{\Sigma}_{\mathbf{b},0}$—the prior covariance of $\mathbf{b}$—need only to be set to a large enough value (e.g., $10^3\mathbf{I}$, where $\mathbf{I}$ is the identity matrix), which corresponds to an uninformative prior on $\mathbf{b}$ (i.e., the probability

distribution is a relatively flat Gaussian). $\mathbf{\Sigma}_{\mathbf{b},0}$ in Eq. (4.4) can be interpreted to be a stabilizing ridge-like value, similar to that of ridge regression, to ensure that the regression does not break down in the presence of collinear input data.

Secondly, $\mathbf{b}_0$ is usually initialized to zero, unless informative prior knowledge is available. As $\mathbf{b}_0$ is multiplied by $\mathbf{\Sigma}_{\mathbf{b},0}^{-1}$, it does not have any real influence on the update equations unless $\mathbf{\Sigma}_{\mathbf{b},0}$ is chosen to be informative.

Thirdly, the prior scale parameters $a_{w_i,0}$ and $b_{w_i,0}$ should be selected so that the weights $\langle w_i \rangle$ are 1 with some confidence. That is to say, we start by assuming that all points are inliers. For example, we can set $a_{w_i,0} = 1$ and $b_{w_i,0} = 1$ so that $\langle w_i \rangle$ has a prior mean of $a_{w_i,0}/b_{w_i,0} = 1$ with a variance of $a_{w_i,0}/b_{w_i,0}^2 = 1$. By using these values, the maximum value of $\langle w_i \rangle$ is capped at 1.5. If the user has good reason to insert strong biases towards particular parameter values (e.g., some prior knowledge on the amount of outliers), then these values should be used. Otherwise, the set of prior parameter values outlined above can be used.

The key point of this Bayesian treatment of weighted regression with heteroscedastic variance is that each data point will be assigned a posterior weight that is indicative of the amount of variance it has, relative to the average variance of the dataset. Consequently, a data point will be downweighted if its variance is much higher than that of the average variance. This algorithm does not require any tuning of threshold values or any user intervention beforehand, performing automatic outlier detection and removal in a black box-like way.

4.2.2 Incremental Version

The algorithm above is suitable if the data D is available in batch form. However, as in most robotic systems, data is often available from sensors one sample at a time, and filtering of the data needs to be done in a real-time, incremental (i.e. online) fashion. Hence, we take the Bayesian weighted model from Eq. (4.2) and modify it to make it an online algorithm. As typical in online algorithms, we introduce a forgetting rate to specify the window over which we wish to average data (Ljung & Soderstrom 1983). We use a scalar forgetting rate, λ, where $0 \leq \lambda \leq 1$, to exponentially discount data collected in the past. The forgetting rate enters the algorithm by accumulating the sufficient statistics of the batch algorithm in an incremental way. The sufficient statistics can be extracted by examining the EM update equations in Eqs. (4.4) to (4.7). As the kth data point becomes available from the sensors, we can calculate the update equations for $\mathbf{b}$ and σ^2 as follows:

$$\boldsymbol{\Sigma}_{\mathbf{b}}^{(k)} = \left(\boldsymbol{\Sigma}_{\mathbf{b},0}^{-1} + \frac{1}{\sigma^2} \text{sum}_k^{wxx^T} \right)^{-1} \tag{4.8}$$

$$\langle \mathbf{b} \rangle^{(k)} = \boldsymbol{\Sigma}_{\mathbf{b}}^{(k)} \left(\boldsymbol{\Sigma}_{\mathbf{b},0}^{-1} \mathbf{b}_0 + \frac{1}{\sigma^2} \text{sum}_k^{wyx} \right) \tag{4.9}$$

$$\begin{aligned} \left(\sigma^2\right)^{(k)} = \frac{1}{N_k} \Big[& \text{sum}_k^{wy^2} - 2\text{sum}_k^{wyx} \langle \mathbf{b} \rangle^{(k)} + \left(\langle \mathbf{b} \rangle^{(k)} \right)^T \text{sum}_k^{wxx^T} \langle \mathbf{b} \rangle^{(k)} \\ & + \mathbf{1}^T \text{diag} \left\{ \text{sum}_k^{wxx^T} \boldsymbol{\Sigma}_{\mathbf{b}}^{(k)} \right\} \Big] \end{aligned} \tag{4.10}$$

where the sufficient statistics, exponentially discounted by λ, are:

$$N_k = 1 + \lambda N_{k-1} \tag{4.11}$$

$$\text{sum}_k^{wxx^T} = \langle w_k \rangle \, \mathbf{x}_k \mathbf{x}_k^T + \lambda \text{sum}_{k-1}^{wxx^T} \tag{4.12}$$

$$\text{sum}_k^{wyx} = \langle w_k \rangle \, y_k \mathbf{x}_k + \lambda \text{sum}_{k-1}^{wyx} \tag{4.13}$$

$$\text{sum}_k^{wy^2} = \langle w_k \rangle \, y_k^2 + \lambda \text{sum}_{k-1}^{wy^2} \tag{4.14}$$

and all of $N_k, \text{sum}_k^{wxx^T}, \text{sum}_k^{wyx}, \text{sum}_k^{wy^2}$ are 0 for $k = 0$.

Notice that the calculation of the posterior covariances of $\mathbf{b}$ in Eqs. (4.4) and (4.8) requires a matrix inversion, resulting in a computational complexity of $O(d^3)$. This will be fine for low-dimensional systems. However, for systems where the data has a large number of input dimensions, the matrix inversion becomes computationally prohibitive. In such situations, Eq. (4.8) can be re-written recursively, as in Recursive Least Squares (Ljung & Soderstrom 1983, Bierman 1977), in order to reduce the computational complexity to $O(d)$ per EM iteration. Given knowledge of the frequency of incoming data, the value of λ can be set accordingly, since the number of data samples that is not "forgotten" is $1/(1-\lambda)$. Additionally, the regression estimates come with a measure of confidence (the posterior covariance of $\mathbf{b}$), such that the quality of the estimates and predictions can be judged.

Naturally, this incremental approximation of the batch Bayesian algorithm comes at a cost since data points that initially appeared to be outliers may actually have been inliers (once we have collected enough data samples to realize this). If the forgetting rate λ used is small enough, then this effect will be less pronounced since the window size of past data samples we are averaging over will be small as well. Hence, if this inlier falls outside the window of the past $1/(1-\lambda)$ data samples, the effect of mistaking an inlier as

an outlier will be less pronounced. At the same time, λ should not be too small in order to ensure that the discrepancy in results between the incremental and batch versions is not too great. This trade-off between preserving equivalency with the batch version and discounting past events is a known issue with the use of forgetting factors for incremental algorithms.

4.3 An Outlier-Robust Kalman Filter

4.3.1 Background

The Kalman filter (Kalman 1960, Kalman & Bucy 1961) is widely used for estimating the state of a dynamic system, given noisy measurement data. It is the optimal *linear* estimator for linear Gaussian systems, giving the minimum mean squared error (Morris 1976). Unlike techniques that require access to the entire set of observed samples, such as the Kalman smoother, e.g., (Jazwinski 1970, Bar-Shalom, Li & Kirubarajan 2001), the Kalman filter assumes that only observations up to the current time step have been observed, making it suitable for real-time tracking. Using state estimates, the filter can also estimate what the corresponding (output) data should be. However, the performance of the Kalman filter degrades when the observed data contains outliers.

To address this, previous work has tried to make the Kalman filter more robust to outliers by addressing the sensitivity of the squared error criterion to outliers (Tukey 1960, Huber 1964). One class of approaches considers non-Gaussian distributions for random variables, e.g., (Sorensen & Alspach 1971, West 1981, West 1982, Smith & West 1983) since multivariate Gaussian distributions are known to be susceptible to outliers.

For example, Meinhold & Singpurwalla (1989) use multivariate Student-t distributions. However, the resulting estimation of parameters may be quite complicated for systems with transient disturbances.

Other efforts have modeled observation and state noise as non-Gaussian, heavy-tailed distributions in order to account for non-Gaussian noise and outliers, e.g., (Masreliez 1975, Masreliez & Martin 1977, Schick & Mitter 1994). These filters are typically more difficult to implement and may no longer provide the conditional mean of the state vector. Other approaches use resampling techniques, e.g., (Kitagawa 1987, Kramer & Sorenson 1988), or numerical integration, e.g., (Kitagawa 1996, Kitagawa & Gersch 1996), but these may require heavy computation not suitable for real-time applications.

Yet another class of methods uses a weighted least squares approach, as done in robust least squares (Ryan 1997, Huber 1973), where the measurement residual error is assigned some statistical property. Some of these algorithms fall under the first category of approaches as well, assuming non-Gaussian distributions for variables. Each data sample is assigned a weight that indicates its contribution to the hidden state estimate at each time step. This technique has been used to produce a Kalman filter that is more robust to outliers, e.g., (Durovic & Kovacevic 1999, Chan, Zhang & Tse 2005). Nevertheless, these methods usually model the weights as some heuristic function of the data, e.g., the Huber function (Huber 1973), and often require tuning or cross-validation of threshold parameters for optimal performance. Using incorrect or inaccurate estimates for the weights may lead to deteriorated performance, so special attention and care is necessary when using these techniques.

In this section, we are interested in making the Kalman filter more robust to the outliers in the observations (i.e. the filter should identify and eliminate possible outliers as it tracks observed data). Estimation of the system dynamics and detection of outliers in the states are different problems and left for future work. We extend the weighted least squares approach of Section 4.2 to the Kalman filter, resulting in a filter that detects outliers in the observations without any parameter tuning or heuristic methods (Ting, Theodorou & Schaal 2007). The filter learns the weights of each data sample and the system dynamics, using an incremental EM framework (Dempster et al. 1977). For ease of analytical computation, we assume Gaussian distributions for variables and states.

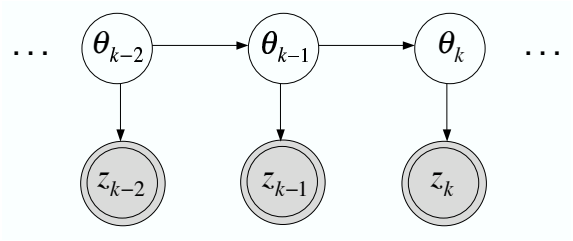
4.3.2 The Kalman Filter

Let us assume we have data observed over N time steps, $\{\mathbf{z}_k\}_{k=1}^N$, and the corresponding hidden states as $\{\boldsymbol{\theta}_k\}_{k=1}^N$, where $\boldsymbol{\theta}_k \in \Re^{d_2 \times 1}, \mathbf{z}_k \in \Re^{d_1 \times 1}$. Assuming a time-invariant system, the Kalman filter system equations are:

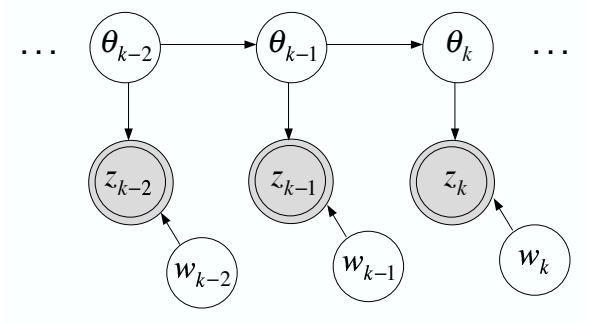
$$\begin{aligned} \mathbf{z}_k &= \mathbf{C}\boldsymbol{\theta}_k + \mathbf{v}_k \\ \boldsymbol{\theta}_k &= \mathbf{A}\boldsymbol{\theta}_{k-1} + \mathbf{s}_k \end{aligned} \tag{4.15}$$

where $\mathbf{C} \in \Re^{d_1 \times d_2}$ is the observation matrix, $\mathbf{A} \in \Re^{d_2 \times d_2}$ is the state transition matrix, $\mathbf{v}_k \in \Re^{d_1 \times 1}$ is the observation noise at time step k, and $\mathbf{s}_k \in \Re^{d_2 \times 1}$ is the state noise at time step k. We assume $\mathbf{v}_k$ and $\mathbf{s}_k$ to be uncorrelated additive mean-zero Gaussian noise: $\mathbf{v}_k \sim \text{Normal}\,(0, \mathbf{R})$, $\mathbf{s}_k \sim \text{Normal}\,(0, \mathbf{Q})$, where $\mathbf{R} \in \Re^{d_1 \times d_1}$ is a diagonal matrix with $\mathbf{r} \in \Re^{d_1 \times 1}$ on its diagonal, and $\mathbf{Q} \in \Re^{d_2 \times d_2}$ is a diagonal matrix with $\mathbf{q} \in \Re^{d_2 \times 1}$

on its diagonal. $\mathbf{R}$ and $\mathbf{Q}$ are covariance matrices for the observation and state noise, respectively.



(a) Kalman filter



(b) Robust weighted Kalman filter

Figure 4.2: Graphical model of Kalman filter and robust weighted Kalman filter. Circular nodes are random variables and shaded double circles are observed random variables. System matrices have been omitted for the sake of graphical clarity.

Figure 4.2(a) shows the graphical model for the standard Kalman filter. Its corresponding filter propagation and update equations are, for $k = 1, .., N$:

Propagation:

$$\boldsymbol{\theta}'_k = \mathbf{A} \langle \boldsymbol{\theta}_{k-1} \rangle \tag{4.16}$$

$$\boldsymbol{\Sigma}'_k = \mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{4.17}$$

Update:

$$\mathbf{S}'_k = \left(\mathbf{C}\boldsymbol{\Sigma}'_k\mathbf{C}^T + \mathbf{R}\right)^{-1} \tag{4.18}$$

$$K'_k = \mathbf{\Sigma}'_k \mathbf{C}^T \mathbf{S}'_k \tag{4.19}$$

$$\langle \boldsymbol{\theta}_k \rangle = \boldsymbol{\theta}'_k + K'_k \left(\mathbf{z}_k - \mathbf{C}\boldsymbol{\theta}'_k \right) \tag{4.20}$$

$$\mathbf{\Sigma}_k = \left(\mathbf{I} - K'_k \mathbf{C} \right) \mathbf{\Sigma}'_k \tag{4.21}$$

where $\langle \boldsymbol{\theta}_k \rangle$ is the posterior mean vector of the state $\boldsymbol{\theta}_k$, $\mathbf{\Sigma}_k$ is the posterior covariance matrix of $\boldsymbol{\theta}_k$, and $\mathbf{S}'_k$ is the covariance matrix of the residual prediction error—all at time step k. In this problem, the system dynamics—$\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$—are unknown, and it is possible to use a maximum likelihood framework to estimate these parameter values (Myers & Tapley 1976). Unfortunately, this standard Kalman filter model considers all data samples to be part of the data cloud and is not robust to outliers.

4.3.3 The Robust Weighted Kalman Filter

To overcome this limitation, we introduce a scalar weight w_k for each observed data sample $\mathbf{z}_k$ such that the variance of $\mathbf{z}_k$ is weighted with w_k, as done in Section 4.2.1. We model the weights to be Gamma distributed random variables, as done previously in Section 4.2.1 for weighted linear regression, and learn estimates for the system dynamics at each time step. A Gamma prior distribution is chosen for the weights in order to ensure they remain positive. Figure 4.2(b) shows the graphical model of the robust weighted Kalman filter. The resulting prior distributions of the random variables are:

$$\begin{aligned} \mathbf{z}_k | \boldsymbol{\theta}_k, w_k &\sim \text{Normal}\left(\mathbf{C}\boldsymbol{\theta}_k, \mathbf{R}/w_k \right) \\ \boldsymbol{\theta}_k | \boldsymbol{\theta}_{k-1} &\sim \text{Normal}\left(\mathbf{A}\boldsymbol{\theta}_{k-1}, \mathbf{Q} \right) \\ w_k &\sim \text{Gamma}\left(a_{w_k}, b_{w_k} \right) \end{aligned} \tag{4.22}$$

Using an incremental EM framework, we can learn the weights of each data sample and system dynamics by maximize the log likelihood $\log p(\mathbf{z}_{1:N})$. We have access to only a lower bound of this measure, based on the expected value of the complete log likelihood—which, until time step k, is given by:

$$\log p(\mathbf{z}_{1:k}, \boldsymbol{\theta}_{0:k}, \mathbf{w}) = \log \left\{ \prod_{i=1}^{k} p(\mathbf{z}_i|\boldsymbol{\theta}_i, w_i) p(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1}) p(\boldsymbol{\theta}_0) \right\} \tag{4.23}$$

The expectation of the complete log likelihood should be taken with respect to the true posterior distribution of all hidden variables $Q(\mathbf{w}, \boldsymbol{\theta})$. Since this is an analytically intractable expression, we use a technique from variational calculus to construct a lower bound and make a factorial approximation of the true posterior as follows: $Q(\mathbf{w}, \boldsymbol{\theta}) = \prod_{i=1}^{K} Q(w_i) \prod_{i=1}^{K} Q(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1}) Q(\boldsymbol{\theta}_0)$. The factorization of $\boldsymbol{\theta}$ considers the influence of each $\boldsymbol{\theta}_i$ from within its Markov blanket, conserving the Markov property that Kalman filters, by definition, have.

While losing a small amount of accuracy, all resulting posterior distributions over hidden variables become analytically tractable. This factorial approximation was chosen purposely so that $Q(w_k)$ is independent from $Q(\boldsymbol{\theta}_k)$; performing joint inference of w_k and $\boldsymbol{\theta}_k$ does not make sense in the context of our generative model. The posterior distribution of the weights $\mathbf{w}$ can then be inferred, in an variational E-step, to be:

$$\langle w_k \rangle = \frac{a_{w_k,0} + \frac{1}{2}}{b_{w_k,0} + \left\langle (\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k)^T \mathbf{R}_k^{-1} (\mathbf{z}_k - \mathbf{C}_k \boldsymbol{\theta}_k) \right\rangle} \tag{4.24}$$

To derive the posterior mean and update of $\boldsymbol{\theta}$, we can first derive the recursive propagation and update equations of the Kalman filter using Bayes' rule and the Markov assumption. The resulting propagation and update equations of the robust weighted Kalman filter are:

Propagation:

$$\boldsymbol{\theta}'_k = \mathbf{A}\left\langle \boldsymbol{\theta}_{k-1} \right\rangle \tag{4.25}$$

$$\boldsymbol{\Sigma}'_k = \mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^T + \mathbf{Q} \tag{4.26}$$

Update:

$$\mathbf{S}'_k = \left(\mathbf{C}\boldsymbol{\Sigma}'_k\mathbf{C}^T + \frac{1}{\langle w_k \rangle}\mathbf{R}\right)^{-1} \tag{4.27}$$

$$K'_k = \boldsymbol{\Sigma}'_k\mathbf{C}^T\mathbf{S}'_k \tag{4.28}$$

$$\left\langle \boldsymbol{\theta}_k \right\rangle = \boldsymbol{\theta}'_k + K'_k\left(\mathbf{z}_k - \mathbf{C}\boldsymbol{\theta}'_k\right) \tag{4.29}$$

$$\boldsymbol{\Sigma}_k = \left(\mathbf{I} - K'_k\mathbf{C}\right)\boldsymbol{\Sigma}'_k \tag{4.30}$$

Eqs. (4.25) to (4.30) can be re-expressed, with a little algebraic manipulation, in terms of the posterior mean and variance for $\boldsymbol{\theta}$:

$$\boldsymbol{\Sigma}_k = \left(\langle w_k \rangle \mathbf{C}_k^T\mathbf{R}_k^{-1}\mathbf{C}_k + \left(\mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^T + \mathbf{Q}_k\right)^{-1}\right)^{-1} \tag{4.31}$$

$$\left\langle \boldsymbol{\theta}_k \right\rangle = \boldsymbol{\Sigma}_k\left(\left(\mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^T + \mathbf{Q}_k\right)^{-1}\mathbf{A}_k\left\langle \boldsymbol{\theta}_{k-1} \right\rangle + \langle w_k \rangle \mathbf{C}_k^T\mathbf{R}_k^{-1}\mathbf{z}_k\right) \tag{4.32}$$

The system dynamics—$\mathbf{C}, \mathbf{A}, \mathbf{R}$ and $\mathbf{Q}$—can be estimated in the M-step to be:

$$\mathbf{C}_k = \left(\sum_{i=1}^{k} \langle w_i \rangle \, \mathbf{z}_i \, \langle \boldsymbol{\theta}_i \rangle^T \right) \left(\sum_{i=1}^{k} \langle w_i \rangle \, \langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \rangle \right)^{-1} \tag{4.33}$$

$$\mathbf{A}_k = \left(\sum_{i=1}^{k} \langle \boldsymbol{\theta}_i \rangle \, \langle \boldsymbol{\theta}_{i-1} \rangle^T \right) \left(\sum_{i=1}^{k} \langle \boldsymbol{\theta}_{i-1} \boldsymbol{\theta}_{i-1}^T \rangle \right)^{-1} \tag{4.34}$$

$$r_{km} = \frac{1}{k} \sum_{i=1}^{k} \langle w_i \rangle \left\langle (\mathbf{z}_{im} - \mathbf{C}_k(m,:)\boldsymbol{\theta}_i)^2 \right\rangle \tag{4.35}$$

$$q_{kn} = \frac{1}{k} \sum_{i=1}^{k} \left\langle (\boldsymbol{\theta}_{in} - \mathbf{A}_k(n,:)\boldsymbol{\theta}_{i-1})^2 \right\rangle \tag{4.36}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$; r_{km} is the mth coefficient of the vector $\mathbf{r}_k$; q_{kn} is the nth coefficient of the vector $\mathbf{q}_k$; $\mathbf{C}_k(m,:)$ is the mth row of the matrix $\mathbf{C}_k$; $\mathbf{A}_k(n,:)$ is the nth row of the matrix $\mathbf{A}_k$; and $a_{w_k,0}$ and $b_{w_k,0}$ are prior scale parameters for the weight w_k.

Since storing sensor data is not possible in real-time applications, Eqs. (4.33) to (4.36)—which require access to all observed data samples up to time step k—need to be re-written using only values observed, calculated or used in the current time step k. We can do this by collecting sufficient statistics in Eq. (4.33) to (4.36) and rewriting them as: (4.36) and rewriting them as:

$$\mathbf{C}_k = \text{sum}_k^{\mathbf{w}\mathbf{z}\boldsymbol{\theta}^T} \left(\text{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \right)^{-1} \tag{4.37}$$

$$\mathbf{A}_k = \text{sum}_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} \left(\text{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \right)^{-1} \tag{4.38}$$

$$r_{km} = \frac{1}{k} \left[\text{sum}_{km}^{\mathbf{w}zz} - 2\mathbf{C}_k(m,:)\text{sum}_{km}^{\mathbf{w}z\boldsymbol{\theta}} + \text{diag} \left\{ \mathbf{C}_k(m,:)\text{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} \mathbf{C}_k(m,:)^T \right\} \right] \tag{4.39}$$

$$q_{kn} = \frac{1}{k} \left[\text{sum}_{kn}^{\theta^2} - 2\mathbf{A}_k(n,:)\text{sum}_{kn}^{\theta\boldsymbol{\theta}'} + \text{diag} \left\{ \mathbf{A}_k(n,:)\text{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} \mathbf{A}_k(n,:)^T \right\} \right] \tag{4.40}$$

where $m = 1, .., d_1$, $n = 1, .., d_2$. The sufficient statistics, which are all a function of values observed, calculated or used in time step k (e.g., $\langle w_k \rangle$, $\mathbf{z}_k$, $\langle \boldsymbol{\theta}_k \rangle$, $\langle \boldsymbol{\theta}_{k-1} \rangle$ etc.) are:

$$\text{sum}_k^{\mathbf{wz}\boldsymbol{\theta}^T} = \langle w_k \rangle \mathbf{z}_k \langle \boldsymbol{\theta}_k \rangle^T + \text{sum}_{k-1}^{\mathbf{wz}\boldsymbol{\theta}^T} \qquad \text{sum}_k^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T} = \langle w_k \rangle \left\langle \boldsymbol{\theta}_k \boldsymbol{\theta}_k^T \right\rangle + \text{sum}_{k-1}^{\mathbf{w}\boldsymbol{\theta}\boldsymbol{\theta}^T}$$

$$\text{sum}_k^{\boldsymbol{\theta}\boldsymbol{\theta}'} = \langle \boldsymbol{\theta}_k \rangle \langle \boldsymbol{\theta}_{k-1} \rangle^T + \text{sum}_{k-1}^{\boldsymbol{\theta}\boldsymbol{\theta}'} \qquad \text{sum}_k^{\boldsymbol{\theta}'\boldsymbol{\theta}'} = \left\langle \boldsymbol{\theta}_{k-1} \boldsymbol{\theta}_{k-1}^T \right\rangle + \text{sum}_{k-1}^{\boldsymbol{\theta}'\boldsymbol{\theta}'}$$

$$\text{sum}_{km}^{\mathbf{w}zz} = \langle w_k \rangle z_{km}^2 + \text{sum}_{k-1}^{\mathbf{w}zz} \qquad \text{sum}_{km}^{\mathbf{w}z\boldsymbol{\theta}} = \langle w_k \rangle z_{km} \boldsymbol{\theta}_k + \text{sum}_{k-1,m}^{\mathbf{w}z\boldsymbol{\theta}}$$

$$\text{sum}_{kn}^{\theta^2} = \left\langle \theta_{kn}^2 \right\rangle + \text{sum}_{k-1,n}^{\theta^2} \qquad \text{sum}_{kn}^{\theta\boldsymbol{\theta}'} = \langle \theta_{kn} \rangle \langle \boldsymbol{\theta}_{k-1} \rangle + \text{sum}_{kn}^{\theta\boldsymbol{\theta}'}$$

Eqs. (4.24) to (4.30) and (4.37) to (4.40) should be computed once for each time step k, e.g., (Ghahramani & Hinton 1996, Neal & Hinton 1999), when the data sample $\mathbf{z}_k$ becomes available.

Initialization of Priors: A few remarks should be made regarding the initialization of priors used in the equations above. In particular, the prior scale parameters $a_{w_k,0}$ and $b_{w_k,0}$ should be selected so that the weights $\langle w_k \rangle$ are 1 with some confidence. That is to say, the algorithm starts by assuming most data samples are inliers. As described in Section 4.2.1, we set $a_{w_k,0} = 1$ and $b_{w_k,0} = 1$ and use these values for any data set unless prior information regarding presence of outliers is available.

If the user has prior knowledge regarding the strong or weak presence of outliers in the data set (and hence, a good reason to insert strong biases towards particular parameter values), the prior scale parameters of the weights can be modified accordingly to reflect this. Since some prior knowledge about the observed data's properties must be known in order to distinguish whether a data sample is an outlier

or part of the data's structure, this Bayesian approach provides a natural framework to incorporate this information.

Secondly, the algorithm is relatively insensitive to the the initialization of $\mathbf{A}$ and $\mathbf{C}$ and will always converge to the same final solution, regardless of these values. For our experiments, we initialize $\mathbf{C} = \mathbf{A} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Finally, the initial values of $\mathbf{R}$ and $\mathbf{Q}$ should be set based on the user's initial estimate of how noisy the observed data is (e.g., $\mathbf{R} = \mathbf{Q} = 0.01\mathbf{I}$ for noisy data, $\mathbf{R} = \mathbf{Q} = 10^{-4}\mathbf{I}$ for less noisy data (Maybeck 1979)).

Outlier detection in the Kalman filter emerges in a similar manner to linear regression. If the prediction error of a data sample $\mathbf{z}_k$ is very large, then the weight $\langle w_k \rangle$ of that data sample will be very small. This leads to a very small $\mathbf{S}'_k$ and small Kalman gain K'_k. In short, the influence of the data sample $\mathbf{z}_k$ will be downweighted when predicting $\boldsymbol{\theta}_k$.

The resulting robust weighted Kalman filter has a computational complexity on the same order as that of a standard Kalman filter since matrix inversions are still needed (for the calculation of covariance matrices). In comparison to other Kalman filters that use heuristics or require more involved implementation, this outlier-robust Kalman filter is principled and easy to implement.

4.3.4 Monitoring the Residual Error

A common sanity check is to monitor the residual error of the data $\mathbf{z}_{1:N}$ and the hidden states $\boldsymbol{\theta}_{1:N}$ in order to ensure that the residual error values stay within the 3σ bounds computed by the filter (Maybeck 1979). If we had access to the true state θ_k for time step k, we would plot the residual state error $(\boldsymbol{\theta}_k - \langle \boldsymbol{\theta}_k \rangle)$ for all time steps k, along with

the corresponding $\pm 3\sigma_k$ values, where $\sigma_k^2 = \text{diag}\{\mathbf{\Sigma}_k\}$. We would also plot the residual prediction error $(\mathbf{z}_k - \mathbf{CA}\langle\boldsymbol{\theta}_{k-1}\rangle)$ for all time steps k, along with the corresponding $\pm 3\sigma_{z_k}$ values, where $\sigma_{z_k}^2 = \text{diag}\{\mathbf{S}'_k\}$.

With these graphs, we should observe the residual error values remaining within the $\pm 3\sigma$ bounds and check that the residual error does not diverge over time. Residual monitoring may be useful to verify that spurious data samples are rejected since processing of these samples may result in corrupted filter computations. It offers a peek into the Kalman filter, providing insights as to how the filter performs.

4.4 Evaluation

4.4.1 Linear Regression

We evaluated our algorithm's ability to automatically detect outliers on a synthetic data set, before implementing it on a robotic dog, LittleDog, manufactured by Boston Dynamics Inc. (Cambridge, MA). We compared it to four other techniques for outlier detection: i) a thresholding approach that classifies a data sample as an outlier if its Mahalanobis distance exceeds an optimal hand-tuned threshold; ii) a 2-component mixture model (a Gaussian distribution to account for inliers and a uniform distribution for outliers); iii) robust least squares regression with bisquare weights (Hoaglin 1983); and iv) Faul & Tipping's variational Bayesian algorithm for robust regression (see Section 4.2.1).

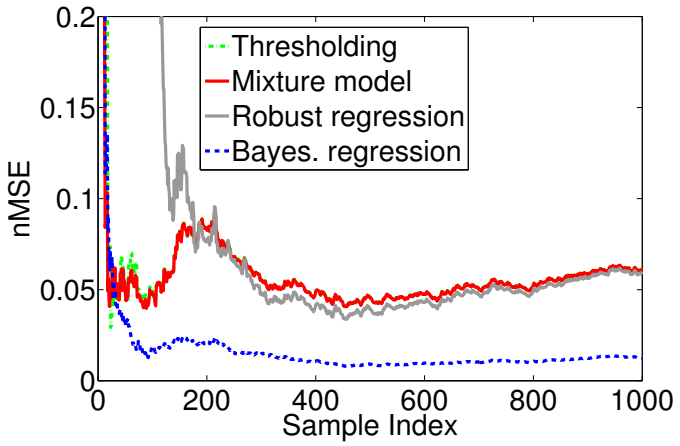
Table 4.1: Average normalized mean squared prediction error for a linear function with 5 input dimensions, evaluated in batch form over 10 trials for all algorithms: σ is the standard deviation of the true conditional output mean and SNR of the outputs is 10.

Algorithm	Distance of outliers from mean		
	$+3\sigma$	$+ 2\sigma$	$+ \sigma$
Thresholding	0.0903	0.0503	0.0232
Mixture Model	0.1327	0.0688	0.0286
Robust Least Squares	0.1890	0.1518	0.0880
Robust Regression	0.1320	0.0683	0.0282
Bayesian weighted regression	0.0273	0.0270	0.0210

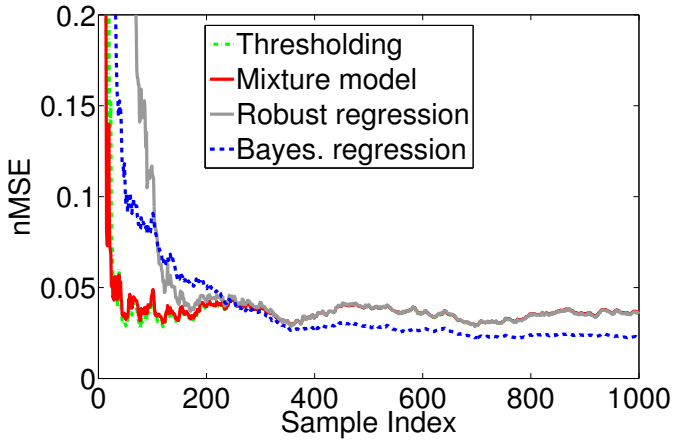
4.4.1.1 Synthetic Data

First, we evaluated all five algorithms on a linear regression problem, where the data is available in batch form. The synthetic data set had 5 input dimensions, 1000 data points (20% of which were outliers), and additive Gaussian noise with a signal-to-noise ratio (SNR) of 10. Outliers were created to be $k\sigma$ away from the true mean of the outputs, where σ is the standard deviation of the true conditional mean of the outputs and k is an integer scaling factor. Table 4.1 shows the average prediction error on noiseless test data for all algorithms. Bayesian weighted linear regression achieves the *lowest* average normalized mean squared error (nMSE). Thresholding works well when the threshold value is hand-tuned optimally, while the remaining methods are less robust to outliers.

Next, the algorithms were evaluated incrementally using a forgetting rate of $\lambda = 0.999$ on the synthetic training data from the first experiment. Robust least squares was omitted since it is a batch algorithm, and making it recursive is non-trivial. Figures 4.3(a) and 4.3(b) track the error in the predicted outputs on the training data. The Bayesian weighted algorithm, shown in the dark blue dotted line, reduces the error to a value that is lowest.



(a) Outliers are at least 3σ from true output mean



(b) Outliers are at least 2σ from true output mean

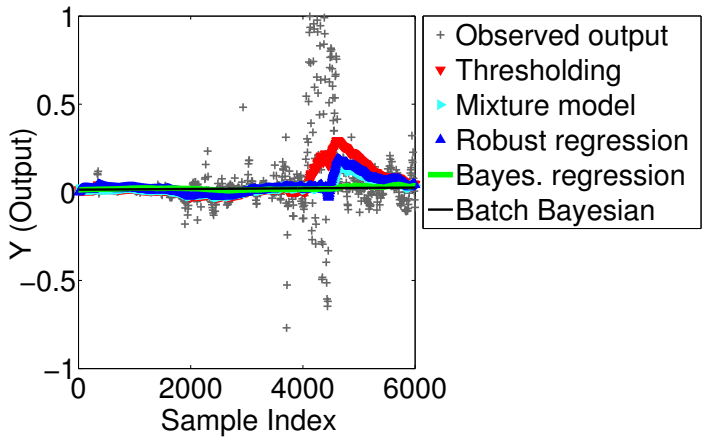
Figure 4.3: Normalized mean squared error values for the synthetic batch data sets used in Table 4.1, evaluated in an incremental manner for thresholding, mixture models, robust regression of Faul & Tipping (2001), and Bayesian weighted regression (Bayes. regression). $\lambda = 0.999$.

Figure 4.4: LittleDog quadruped robot by Boston Dynamics (Cambridge, MA).

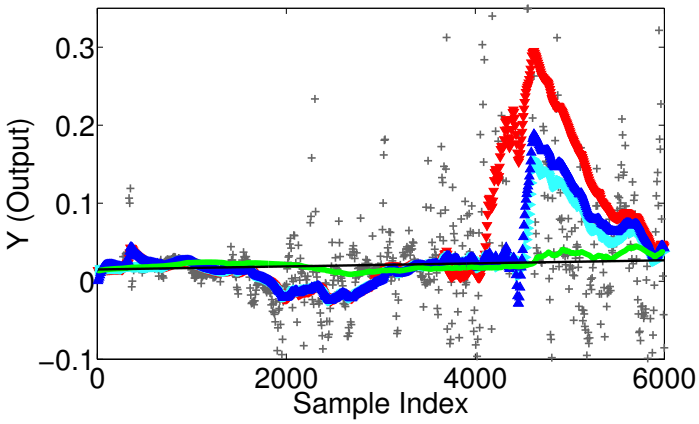
4.4.1.2 LittleDog Robot

We evaluated the algorithms on a 12 degree-of-freedom (DOF) robotic dog, LittleDog, as shown in Figure 4.4. The dog has two sources that measure its orientation: the motion capture (MOCAP) system and an on-board inertia measurement unit (IMU). Both provide a quaternion q of the robot's orientation: q_{MOCAP} from the MOCAP and q_{IMU} from the IMU. q_{IMU} drifts over time since the IMU cannot provide stable orientation estimation but its signal is clean. q_{MOCAP} has outliers and noise, but no drift. We want to estimate the offset between q_{MOCAP} and q_{IMU}, and this offset is a noisy, outlier-infested, slowly drifting signal.

Figure 4.5(a) shows the offset data between q_{MOCAP} and q_{IMU} for one of the four quaternion coefficients, collected over 6000 data samples. The thresholding, mixture model and variational Bayesian robust regression approaches appear sensitive to outliers (occurring between the 4000th and 5000th sample). In comparison, the incremental version of Bayesian weighted regression is far less sensitive to outliers.



(a) Predicted versus observed outputs



(b) Magnified view of Figure 4.5(a)

Figure 4.5: Predicted versus observed offset data between the q_{IMU} and q_{MOCAP}, shown for one of the four quaternion coefficients ($\lambda = 0.999$). Observed outputs are noisy and contain outliers.

4.4.2 Weighted Kalman Filter

We evaluated our robust weighted Kalman filter on synthetic and robotic data sets and compared it with three other filters. We omitted the filters of Durovic & Kovacevic (1999) and Chan et al. (2005) since we had difficulty implementing them and getting them to work. Instead, we used a hand-tuned thresholded Kalman filter to serve as a baseline comparison. We compared our robust weighted Kalman filter to two other filters: i) the standard Kalman filter and ii) a Kalman filter where outliers are determined by thresholding on the Mahalanobis distance. This threshold value is manually hand-tuned for a particular data set. If the Mahalanobis distance is less than a certain threshold value, then it is considered an inlier and processed. Otherwise, it is an outlier and ignored. This threshold value is *hand-tuned manually* in order to find the optimal value for a particular data set. If we have *a priori* access to the entire data set and are able to tune this threshold value accordingly, the thresholded Kalman filter gives *near-optimal* performance.

First, we simulate a real data set where hidden states are unknown and only access to observed data is available. Although they are linear, Kalman filters are commonly used to track more interesting "nonlinear" behaviors (i.e., not just a straight line). For this reason, we try the methods on a synthetic data set exhibiting nonlinear behavior, where the system dynamics are unknown. We also conducted experiments on a synthetic data set where the system dynamics of the generative model are known. These experiments yield similar performance results to that where the system dynamics are unknown. Finally, we run all Kalman filters on data collected from LittleDog.

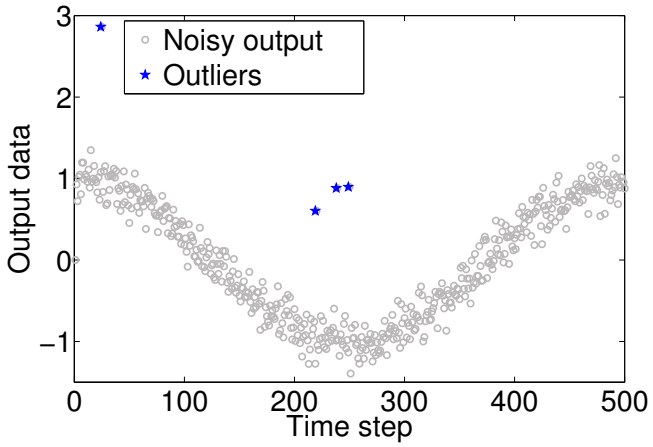
For these experiments, we are interested in the Kalman filter's prediction of the observed (output) data and detection of outliers in the observations. We are not interested in the estimation of the system dynamics or in the estimation (or outlier detection) of the states. Estimation of the system matrices for the purpose of parameter identification and detection of outliers in the states are different problems and left to another paper.

4.4.2.1 Synthetic Data

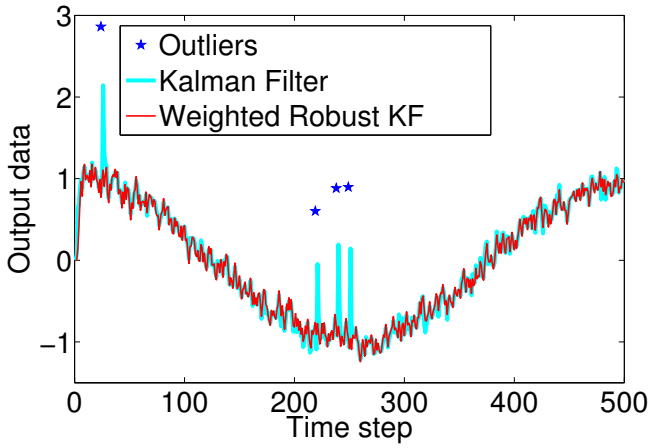
We created data exhibiting nonlinear behavior, where $\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$, $\mathbf{Q}$ and states are unknown, high noise is added to the (output) data, and a data sample is an outlier with 1% probability. One-dimensional data is used for ease of visualization, and Figure 4.6(a) shows a noisy cosine function with outliers, over 500 time steps. For optimal performance, $\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$ were manually tuned for the standard Kalman filter—a tricky and time-consuming process. In contrast, the system dynamics were learnt for the thresholded Kalman filter using a maximum likelihood framework (i.e. using Eqs.(4.37) to (4.40) without any weights).

Figure 4.6(b) shows how sensitive the standard Kalman filter is to outliers, while the weighted robust Kalman filter seems to detect them quite well. In Figure 4.7(a), we compare the weighted robust Kalman filter with thresholded filter. Both filters appear to perform as well, which is unsurprising, given the amount of manual tuning required by the thresholded Kalman filter.

Figure 4.7(b) shows that the residual prediction error on the outputs stays within the $\pm 3\sigma$ bounds. Graphs showing the estimated states were omitted, but they show similar trends in the accuracy results.

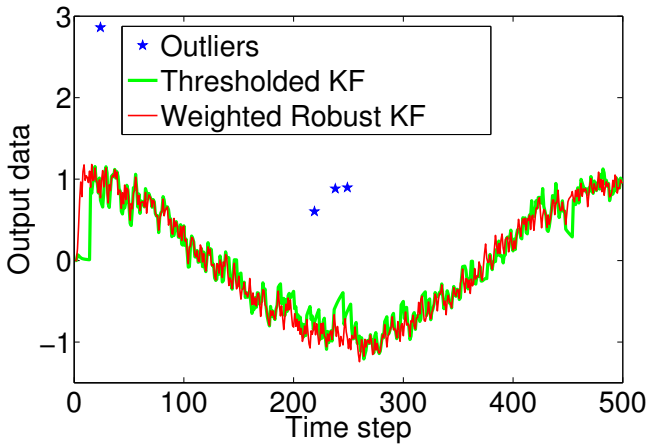


(a) Observed noisy output data with outliers

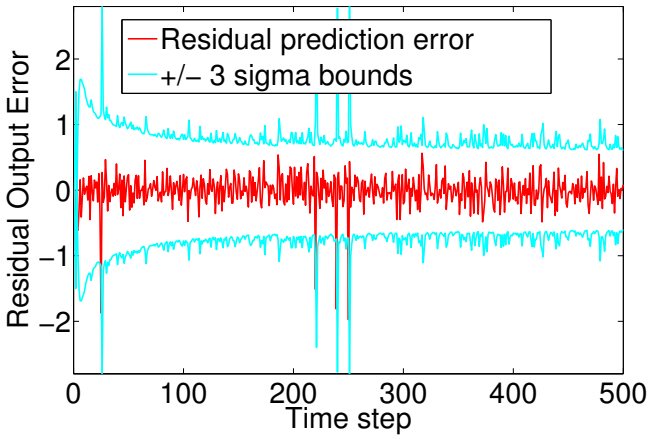


(b) Predicted data for the Kalman filter (KF) and the weighted robust KF

Figure 4.6: One-dimensional data showing a cosine function with noise and outliers (and unknown system dynamics) for 500 samples at 1 sample/time step



(a) Predicted data for the thresholded KF, alternative KF and weighted robust KF. All perform similarly



(b) Residual prediction error for the weighted robust Kalman filter

Figure 4.7: One-dimensional data showing a cosine function with noise and outliers (and unknown system dynamics) for 500 samples at 1 sample/time step

4.4.2.2 LittleDog Robot

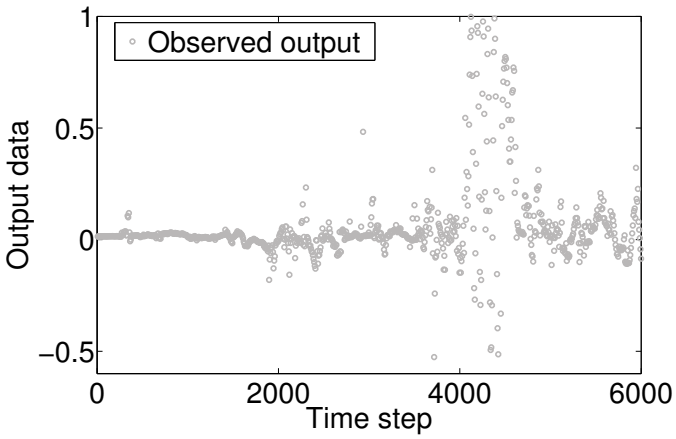
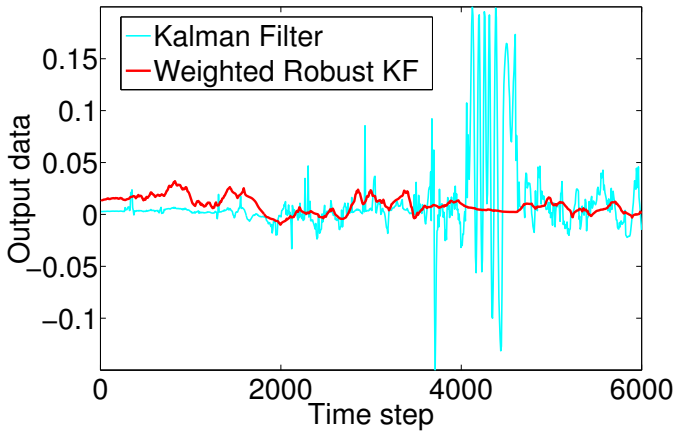


Figure 4.8: Observed quaternion data from LittleDog robot: a slowly drifting noisy signal with outliers. Data is shown for 6000 samples. We assume that each data sample is made available at each time step.

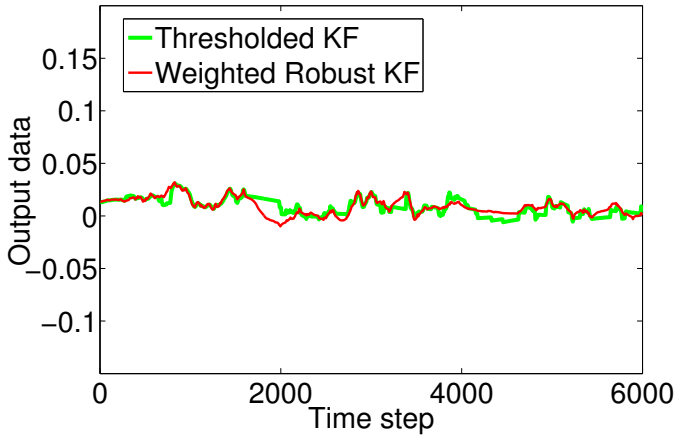
We present the tracking performance of the filters on quaternion data from LittleDog, described previously in Section 4.4.1.2.

Figure 4.8 shows the quaternion data over 6000 data samples, at 1 sample/time step. There are various approaches to estimating this slowly drifting signal, depending on the quality of estimate desired. We can estimate it with a straight line, as done in Section 4.4.1.2. However, if we want to estimate this slowly drifting signal more accurately, we can use the proposed outlier-robust Kalman filter to track it. For optimal performance, we, once again, manually tuned $\mathbf{C}$, $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{Q}$ for the standard Kalman filter. The system dynamics of the thresholded Kalman filter were learnt, and its threshold parameter was manually tuned for best performance on this data set.

The standard Kalman filter fails to detect outliers occurring between the 4000th and 5000th sample, as seen in Figure 4.9(a). Figure 4.9(b) shows that the thresholded Kalman



(a) Predicted quaternion data for the Kalman filter and robust weighted Kalman filter



(b) Predicted quaternion data for the thresholded Kalman filter and robust weighted Kalman filter

Figure 4.9: Predicted quaternion data for the Kalman filter and robust weighted Kalman filter, shown for 6000 samples. We assume that each data sample is made available at each time step. Note the change of scale in axis from Figure 4.8.

filter does not react as violently as the standard Kalman filter to outliers and, in fact, appears to perform as well as the robust weighted Kalman filter. This is to be expected, given that we hand-tuned the threshold parameter for optimal performance.

In this experiment, the advantages offered by the weighted Kalman filter are clear. It outperforms the traditional Kalman filter, while achieving a level of performance on par with a thresholded Kalman filter (where the threshold value is manually tuned for optimal performance).

4.5 Discussion

We introduced a Bayesian weighted regression algorithm that automatically detects and eliminates outliers in real-time. We extended this approach to produce a Kalman filter that is able to track and detect outliers in the observations. Both methods are easy to use and do not require any parameter tuning, interference from the user, heuristics or sampling. Both learn the value of weights associated with the data samples, but they also require an initial weight prior be set (that is, an outlier prior that indicates how strong the presence of outliers are in the data). However, in order to perform outlier detection correctly, this prior is necessary in order to distinguish outliers from structure in the data. The Bayesian weighted regression algorithm outperforms standard methods in batch or incremental settings. The outlier-robust Kalman filter performs as well as a "hand-tuned" Kalman filter that requires *a priori* knowledge of the entire data.

The calculation of the posterior covariances of $\mathbf{b}$ in Eqs. (4.4) and (4.8) requires a matrix inversion, resulting in a computational complexity of $O(d^3)$. This will be fine

for low-dimensional systems. However, for systems where the data has a large number of input dimensions, the matrix inversion becomes computationally prohibitive. In such situations, Eq. (4.8) can be re-written recursively, as in Recursive Least Squares (Ljung & Soderstrom 1983, Bierman 1977), in order to reduce the computational complexity to $O(d)$ per EM iteration.

The regression estimates come with a measure of confidence, such that the quality of the estimates and predictions can be judged. Notice that the calculation of the posterior covariances of $\mathbf{b}$ in Eqs. (4.4) and (4.8) requires a matrix inversion, resulting in a computational complexity of $O(d^3)$. This will be fine for low-dimensional systems. However, for systems where the data has a large number of input dimensions, the matrix inversion becomes computationally prohibitive. In such situations, Eq. (4.8) can be re-written recursively, as in Recursive Least Squares (Ljung & Soderstrom 1983, Bierman 1977), in order to reduce the computational complexity to $O(d)$ per EM iteration.

Chapter 5

Nonlinear High-Dimensional Regression

Having addressed the problem of noisy high-dimensional linear regression, we now move to nonlinear high-dimensional regression problems. Gaussian processes (Bernardo & Smith 1994, Williams & Rasmussen 1995, Rasmussen 1996) (GPs) are competitive function approximators for nonlinear regression problems. However, they can be computationally prohibitive for large data sets since they require a matrix inversion that takes around $O(N^3)$, where N is the number of data samples. Additionally, they are not suitable for high-dimensional ill-conditioned problems (where many of the input dimensions are redundant or irrelevant). Although recent work has tried to overcome these limitations, e.g., (Csato & Opper 2002, Lawrence, Seeger & Herbrich 2003, Y. Shen & Seeger 2006, Snelson & Ghahramani 2006*a*, Snelson & Ghahramani 2006*c*), Gaussian process regression is not quite ready for real-time robot learning applications, where the input data is high-dimensional and fast incremental learning is necessary. In such scenarios, local methods such as Locally Weighted Projection Regression (LWPR) (Vijayakumar et al. 2005) may be more suitable and have been shown to be effective for real-time learning of motor skills on humanoid robots.

We could consider how one would perform Bayesian locally weighted regression in a batch setting (i.e. the entire data set is available at the outset), where the locality of each model is learnt probabilistically. In Schaal & Atkeson (1994), locally weighted learning was applied to robotic devilsticking, and the locality of each model was determined by incremental gradient descent based on stochastic leave-one-out cross-validation.

We could also move beyond locally weighted regression and consider the more general category of kernel-based methods, which include Parzen windows, kernel regression, locally weighted regression, radial basis function networks, Reproducing Kernel Hilbert Spaces, Support Vector Machines, and Gaussian process regression. Most algorithms start with parameterizations that are the same for all kernels, independent of where in data space the kernel is used, but later recognize the advantage of locally adaptive kernels (Friedman 1984, Poggio & Girosi 1990, Fan & Gijbels 1996). Such locally adaptive kernels are useful in scenarios where the data characteristics vary greatly in different parts of the workspace (e.g., in terms of data density, curvature and output noise). For instance, in Gaussian process (GP) regression, using a nonstationary covariance function, e.g., (Paciorek & Schervish 2004), allows for such a treatment. Performing optimizations individually for every kernel, however, becomes rather complex and is prone to overfitting due to a flood of open parameters. Previous work has suggested gradient descent techniques with cross-validation methods or involved statistical hypothesis testing for optimizing the shape and size of a kernel in a learning system (Fan & Gijbels 1992, Fan & Gijbels 1995, Schaal & Atkeson 1994, Friedman 1984).

In this chapter, we consider local kernel shaping by averaging over data samples with the help of locally polynomial models and formulate this approach, in a Bayesian framework, for function approximation with both piecewise linear models and nonstationary GP regression. Our local kernel shaping algorithm (Ting, Kalakrishnan, Vijayakumar & Schaal 2008) is computationally efficient—capable of handling large data sets, can deal with functions of strongly varying curvature, data density and output noise, and even rejects outliers automatically. The algorithm automatically learns the size of the data neighborhood contributing to each local model (i.e. the "bandwidth" or spatial distance metric of the local model). A Bayesian approach offers error bounds on the distance metric and incorporates this uncertainty in the predictive distributions.

Our approach to nonstationary GP regression differs from previous work by avoiding Markov Chain Monte Carlo (MCMC) sampling (Rasmussen & Ghahramani 2002, Meeds & Osindero 2005) and by exploiting the full nonparametric characteristics of GPs in order to accommodate nonstationary data. Other approaches to kernel shaping include the random varying coefficient model (Longford 1993), local polynomial modeling (Fan & Gijbels 1996), use of asympototic analysis (Schucany 1995), and entropy-based measures (Ormoneit & Hastie 1999)—where the volume instead of the bandwidth of the kernel is optimized, to name a few. Many of these methods, however, require cross-validation or are sensitive to the initialization of parameters.

One of the core application domains for our work is learning control, where computationally efficient function approximation and highly accurate local linearizations from data are crucial for deriving controllers and for optimizing control along trajectories (Atkeson & Schaal 1997). The high variations from fitting noise, seen in Figures 5.4(a) and 5.5(a)

are harmful to the learning system, potentially causing the controller to be unstable. Our final evaluations illustrate such a scenario by learning an inverse kinematics model for a real robot arm.

5.1 Bayesian Local Kernel Shaping

We develop our approach in the context of nonparametric locally weighted regression with locally linear polynomials (Atkeson et al. 1997), assuming, for notational simplicity, only a one-dimensional output—extensions to multi-output settings are straightforward. We assume a training set of N samples, $D = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, drawn from a nonlinear function:

$$y = f(\mathbf{x}) + \epsilon$$

that is contaminated with mean-zero (but potentially heteroscedastic) noise ϵ. Each data sample consists of a d-dimensional input vector $\mathbf{x}_i$ and an output y_i. We wish to approximate a locally linear model of this function at a query point $\mathbf{x}_q \in \Re^{d \times 1}$ in order to make a prediction $y_q = \mathbf{b}^T \mathbf{x}^q$, where $\mathbf{b} \in \Re^{d \times 1}$.

We assume the existence of a spatially localized weighting kernel:

$$w_i = K\left(\mathbf{x}_i, \mathbf{x}_q, \mathbf{h}\right)$$

that assigns a scalar weight to every $\{\mathbf{x}_i, y_i\}$ according to its Euclidean distance in input space from the query point $\mathbf{x}_q$. A popular choice of the function K is the Gaussian kernel:

$$w_i = \exp\left\{-0.5\left(\mathbf{x}_i - \mathbf{x}_q\right)^T \mathbf{H}\left(\mathbf{x}_i - \mathbf{x}_q\right)\right\}$$

(where $\mathbf{H}$ is a positive semi-definite diagonal matrix with $\mathbf{h}$ on its diagonal) since the weight of a data sample is then lower when its input point is further away from the query input $\mathbf{x}_q$. That is to say, we assume that the further away a training data sample is from the query point in input space, the more we downweight that training sample. However, other kernels may be used as well (Atkeson et al. 1997).

The bandwidth $\mathbf{h} \in \Re^{d\times 1}$ of the kernel represents how wide the weighting kernel is and dictates the quality of fit of the local model. $\mathbf{h}$ is a form of distance metric, a measure that determines the size and shape of the weighting kernel and is the size of the local regime in input space to be linearized. $\mathbf{h}$ should be chosen as a function of the local curvature of $f(\mathbf{x})$ and the data density around $\mathbf{x}_q$. If we find the "right" bandwidth as a function of $\mathbf{x}_q$ to avoid oversmoothing or overfitting the data, nonlinear function approximation can be solved accurately and efficiently. Our goal is to find a Bayesian formulation of determining $\mathbf{b}$ and $\mathbf{h}$ simultaneously.

As previously mentioned, past work has involved use of cross-validation, involved statistical hypothesis testing or search to find the optimal distance metric value. However, these methods may be sensitive to initialization values (for gradient descent), require manual meta-parameter tuning or be quite computationally involved. In the next section,

we propose a variational Bayesian algorithm that learns both $\mathbf{b}$ and $\mathbf{h}$ simultaneously in an EM-like framework.

5.1.1 Model

For the locally linear model at the query point $\mathbf{x}_q$, we can introduce hidden random variables $\mathbf{z}$ (D'Souza et al. 2004) and modify the linear model $y_i = \mathbf{b}^T \mathbf{x}_i$ so that:

$$\begin{aligned} y_i &= \sum_{m=1}^{d} z_{im} + \epsilon \\ z_{im} &= \mathbf{b}_m^T \mathbf{x}_{im} + \epsilon_{zm} \end{aligned} \tag{5.1}$$

where ϵ_{zm} and ϵ are both additive noise terms:

$$\epsilon \sim \text{Normal}\left(0, \sigma^2\right)$$

$$\epsilon_{zm} \sim \text{Normal}\left(0, \psi_{zm}\right)$$

The $\mathbf{z}$ variables allow us to derive computationally efficient and numerically robust $O(d)$ EM-like updates, as we will see later. Note that $\mathbf{x}_{im} = [x_{im}\ 1]^T$ and $\mathbf{b}_m = [b_m\ b_{m0}]^T$, where x_{im} is the mth coefficient of $\mathbf{x}_i$, b_m is the mth coefficient of $\mathbf{b}$ and b_{m0} is the offset value.

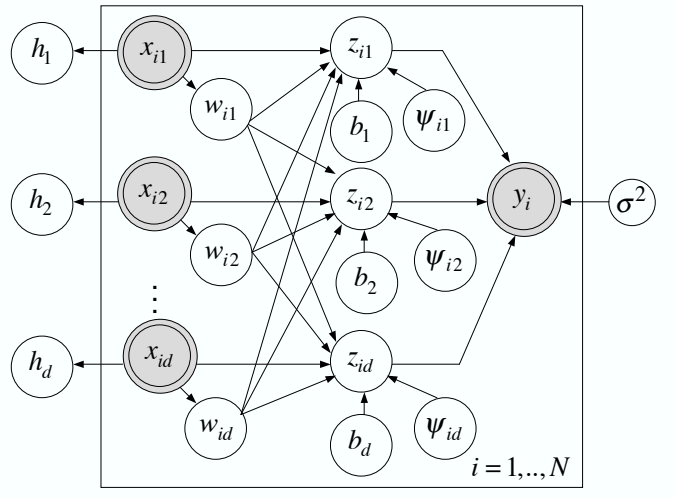


Figure 5.1: Graphical model of Bayesian local kernel shaping. Random variables are in circles, and observed random variables are in shaded double circles.

The prediction at the query point $\mathbf{x}_q$ is then $\sum_m^d \mathbf{b}_m^T \mathbf{x}_{im}$. We assume the following prior distributions for our model, shown graphically in Figure 5.1:

$$\begin{aligned} p(y_i|\mathbf{z}_i) &\sim \text{Normal}\left(\mathbf{1}^T\mathbf{z}_i, \sigma^2\right) \\ p(z_{im}|\mathbf{x}_{im}) &\sim \text{Normal}\left(\mathbf{b}_m^T\mathbf{x}_{im}, \psi_{zm}\right) \\ p(\mathbf{b}_m|\psi_{zm}) &\sim \text{Normal}\left(0, \psi_{zm}\boldsymbol{\Sigma}_{\mathbf{b}_m,0}\right) \\ p(\psi_{zm}) &\sim \text{Scaled-Inv-}\chi^2\left(n_{m0}, \psi_{zm,0}\right) \end{aligned} \tag{5.2}$$

where $\mathbf{1}$ is a vector of 1s, $\mathbf{z}_i \in \Re^{d\times 1}$, z_{im} is the mth coefficient of $\mathbf{z}_i$, and $\boldsymbol{\Sigma}_{\mathbf{b}_m,0}$ is the prior covariance matrix of $\mathbf{b}_m$ and a 2×2 diagonal matrix. n_{m0} and σ^2_{mN0} are the prior parameters of the Scaled-inverse-χ^2 distribution[1]. The Scaled-Inverse-χ^2 distribution was used for ψ_{zm} since it is the conjugate prior for the variance parameter of a Gaussian distribution.

[1] n_{m0} is the number of degrees of freedom parameter and σ^2_{mN0} is the scale parameter

In contrast to classical treatments of Bayesian weighted regression (Gelman et al. 2000) where the weights enter as a heteroscedastic correction on the noise variance of each data sample, we **associate a scalar indicator-like weight, $w_i \in \{0, 1\}$, with each sample** $\{\mathbf{x}_i, y_i\}$ in D. The sample is fully included in the local model if $w_i = 1$ and excluded if $w_i = 0$. We define the weight w_i to be:

$$w_i = \prod_{m=1}^{d} w_{im} \tag{5.3}$$

where w_{im} is the weight component in the mth input dimension. While previous methods model the weighting kernel K as some explicit function, we treat the weights w_{im} probabilistically and model them as Bernoulli-distributed random variables:

$$p(w_{im}) \sim \text{Bernoulli}(q_{im}) \tag{5.4}$$

choosing a symmetric bell-shaped function for the parameter q_{im}:

$$q_{im} = \frac{1}{1 + (x_{im} - x_{qm})^{2r} h_m} \tag{5.5}$$

where x_{qm} is the mth coefficient of $\mathbf{x}_q$, h_m is the mth coefficient of $\mathbf{h}$, and $r > 0$ is a positive integer. The function for q_{im} has the property that data samples are downweighted more if they are further located from the query point in input space. $(x_{im} - x_{qm})$ is taken to the power $2r$ in order to ensure that the resulting expression is positive. Adjusting r affects how long the tails of the kernel are, as Figure 5.2 shows. For smaller values of

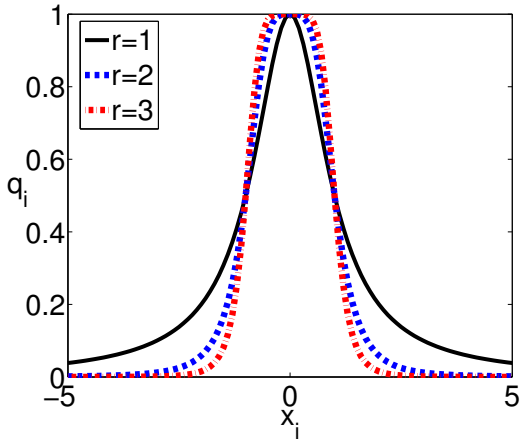


Figure 5.2: Graphs of the function $q_i = \frac{1}{1+(x_i-x_q)^{2r}h}$ for 1-dimensional inputs x_i, where $x_q = 0$ and $h = 1$, over different r values.

r, the weighting kernel takes on a shape with a narrower peak, but longer tails. We use $r = 2$ for all our experiments to avoid long tails.

We model w_{im} with a Bernoulli distribution since the weights $\mathbf{w}$ need to be positive and between 0 and 1. q_{im} is then the parameter of the Bernoulli distribution and is, by definition, also restricted to values between 0 and 1.

As pointed out in Atkeson et al. (1997), the particular mathematical formulation of a weighting kernel is largely computationally irrelevant for locally weighted learning. Our choice of weighting kernel (i.e., choice of function for q_{im}) was dominated by the desire to obtain analytically tractable learning updates (see the next section for more details). We place a Gamma prior over the bandwidth h_m:

$$p(h_m) \sim \text{Gamma}\,(a_{hm0}, b_{hm0}) \tag{5.6}$$

where a_{hm0} and b_{hm0} are parameters of the Gamma distribution, to ensure that a positive weighting kernel width.

5.1.2 Inference

We can treat the entire regression problem as an EM learning problem (Dempster et al. 1977, Ghahramani & Beal 2000). Defining $\mathbf{X}$ to be a matrix with input vectors $\mathbf{x}_i$ arranged in its rows and $\mathbf{y}$ as a vector with coefficients y_i, we would like to maximize the log likelihood $\log p(\mathbf{y}|\mathbf{X})$ for generating the observed data. We can maximize this incomplete log likelihood by maximizing the expected value of the complete log likelihood, shown below:

$$\mathbf{y}, \mathbf{Z}, \mathbf{b}, \mathbf{w}, \mathbf{h}, \sigma^2, \psi_z|\mathbf{X}) = \prod_{i=1}^{N} p(y_i, \mathbf{z}_i, \mathbf{b}, w_i, \mathbf{h}, \sigma^2, \psi_z|\mathbf{x}_i)$$

In our model, each data sample i has an indicator-like scalar weight w_i associated with it, allowing us to express the complete log likelihood L, in a similar fashion to mixture models, as:

$$L = \log \left[\prod_{i=1}^{N} \left[\left[p(y_i|\mathbf{z}_i, \sigma^2) p(\mathbf{z}_i|\mathbf{x}_i, \mathbf{b}, \psi_z) \right]^{w_i} \prod_{m=1}^{d} p(w_{im}) \right] \prod_{m=1}^{d} p(\mathbf{b}_m|\psi_{zm}) p(\psi_{zm}) p(h_m) p(\sigma^2) \right]$$

which can be expanded to:

$$\begin{aligned} L = &\sum_{i=1}^{N} w_i \log p(y_i|\mathbf{z}_i, \sigma^2) + \sum_{i=1}^{N} w_i \log p(\mathbf{z}_i, |\mathbf{x}_i, \mathbf{b}, \psi_z) + \sum_{i=1}^{N} \sum_{m=1}^{d} \log p(w_{im}) \\ &+ \sum_{m=1}^{d} \log p(\mathbf{b}_m|\psi_{zm}) + \sum_{m=1}^{d} \log p(\psi_{zm}) + \sum_{m=1}^{d} \log p(h_m) + \log p(\sigma^2) \end{aligned} \tag{5.7}$$

If we expand the term $\sum_{i=1}^{N}\sum_{m=1}^{d}\log p(w_{im})$ in Eq. (5.7) above, we get:

$$\begin{aligned}
&\sum_{i=1}^{N}\sum_{m=1}^{d}\log p(w_{im}=1)^{w_{im}}+\sum_{i=1}^{N}\sum_{m=1}^{d}\log p(w_{im}=0)^{(1-w_{im})}\\
&=\sum_{i=1}^{N}\sum_{m=1}^{d}\left[w_{im}\log\frac{1}{1+(x_{im}-x_{qm})^{2r}\,h_m}+(1-w_{im})\log\left(1-\frac{1}{1+(x_{im}-x_{qm})^{2r}\,h_m}\right)\right]\\
&=\sum_{i=1}^{N}\sum_{m=1}^{d}w_{im}\log\frac{1}{1+(x_{im}-x_{qm})^{2r}\,h_m}+\sum_{i=1}^{N}\sum_{m=1}^{d}(1-w_{im})\log\frac{(x_{im}-x_{qm})^{2r}\,h_m}{1+(x_{im}-x_{qm})^{2r}\,h_m}\\
&=\sum_{i=1}^{N}\sum_{m=1}^{d}(1-w_{im})\log(x_{im}-x_{qm})^{2r}\,h_m-\sum_{i=1}^{N}\sum_{m=1}^{d}\log\Big(1+(x_{im}-x_{qm})^{2r}\,h_m\Big) \quad (5.8)
\end{aligned}$$

Given that the prior on h_m is a Gamma distribution, there is a problematic log term, $-\log\Big(1+(x_{im}-x_{qm})^{2r}\,h_m\Big)$, in the expression above that prevents us from deriving an analytically tractable expression for the posterior of h_m. To address this, we use a variational approach on concave/convex functions suggested by Jaakkola & Jordan (2000) in order to produce analytically tractable expressions. We can find a lower bound to the term $-\log\Big(1+(x_{im}-x_{qm})^{2r}\,hm\Big)$ so that:

$$-\log\Big(1+(x_{im}-x_{qm})^{2r}\,h_m\Big)\geq-\lambda_{im}\,(x_{im}-x_{qm})^{2r}\,h_m+\text{const}_{h_m} \quad (5.9)$$

where λ_{im} is a variational parameter to be optimized in an M-step of our final EM-like algorithm and can be done so by optimizing with respect to the expected complete log likelihood:

$$\lambda_{im}=\frac{1}{1+(x_{im}-x_{qm})^{2r}\,\langle h_m\rangle} \quad (5.10)$$

Please refer to Appendix D.1 for derivations of the lower bound in Eq. (5.9).

Our choice of weighting kernel q_{im} allows us to find a lower bound to L in this manner. We explored the use of other weighting kernels (such as a quadratic negative exponential, among others). However, we encountered difficulties in finding a lower bound to the problematic terms in $\log p(w_{im})$ so that analytically tractable inference for the posterior of h_m could be done.

The resulting lower bound to L, $\hat{L}$, is then:

$$\begin{aligned}
L \geq \hat{L} = & -\sum_{i=1}^{N} w_i \log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\sum_{i=1}^{N} w_i \left(y_i - \mathbf{1}^T \mathbf{z}_i\right)^2 - \frac{1}{2}\sum_{i=1}^{N} w_i \sum_{m=1}^{d} \log 2\pi\psi_{zm} \\
& - \sum_{i=1}^{N} w_i \sum_{m=1}^{d} \frac{1}{2\psi_{zm}} \left(z_{im} - \mathbf{b}_m^T \mathbf{x}_{im}\right)^2 + \sum_{i=1}^{N}\sum_{m=1}^{d} (1 - w_{im}) \log \left(x_{im} - x_{qm}\right)^{2r} h_m \\
& - \sum_{i=1}^{N}\sum_{m=1}^{d} \lambda_{im} \left(x_{im} - x_{qm}\right)^{2r} h_m - \frac{1}{2}\sum_{m=1}^{d} \log |\mathbf{\Sigma}_{\mathbf{b}_m,0}| - \frac{1}{2}\sum_{m=1}^{d} \log \psi_{zm} \\
& - \sum_{m=1}^{d} \frac{1}{2\psi_{zm}} \mathbf{b}_m^T \mathbf{\Sigma}_{\mathbf{b}_m,0}^{-1} \mathbf{b}_m - \sum_{m=1}^{d} \left(\frac{n_{m0}}{2} + 1\right) \log \psi_{zm} - \sum_{m=1}^{d} \frac{n_{m0}\psi_{zmN0}}{2\psi_{zm}} \\
& + \sum_{m=1}^{d} (a_{hm0} - 1) \log h_m - \sum_{m=1}^{d} b_{hm0} h_m + \text{const}_{\mathbf{y},\mathbf{Z},\mathbf{b},\mathbf{w},\mathbf{h},\sigma^2,\psi_z}
\end{aligned} \tag{5.11}$$

We would like to maximize the lower bound to the log likelihood in Eq. (5.11) and find the parameter values that correspond to this. The expectation of this complete data likelihood $\hat{L}$ should be taken with respect to the true posterior distribution of all hidden variables $Q(\mathbf{b}, \psi_z, \mathbf{z}, \mathbf{h})$. Since this is an analytically tractable expression, a lower bound can be formulated using a variational factorial approximation of the true posterior, e.g., (Ghahramani & Beal 2000), as follows:

$$Q(\mathbf{b}, \psi_z, \mathbf{h}, \mathbf{z}) = Q(\mathbf{b}, \psi_z) Q(\mathbf{h}) Q(\mathbf{z}).$$

All resulting posterior distributions over hidden variables become analytically tractable. Assuming that $\boldsymbol{\theta} = \{\mathbf{b}, \psi_z, \mathbf{h}\}$, the posterior of w_{im}, $p(w_{im} = 1|y_i, \mathbf{z}_i, \mathbf{x}_i, \boldsymbol{\theta}, w_{i,k\neq m})$ can be inferred using Bayes' rule:

$$\frac{p(y_i, \mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}, w_{i,k\neq m}, w_{im} = 1)^{\prod_{t=1,t\neq m}^{d}\langle w_{it}\rangle} p(w_{im} = 1)}{p(y_i, \mathbf{z}_i|\mathbf{x}_i, \boldsymbol{\theta}, w_{i,k\neq m}, w_{im} = 1)^{\prod_{t=1,t\neq m}^{d}\langle w_{it}\rangle} p(w_{im} = 1) + p(w_{im} = 0)} \tag{5.12}$$

where $w_{i,k\neq m}$ denotes the set of weights $\{w_{ik}\}_{k=1,k\neq m}^{d}$. For the dimension m, we account for the effect of weights in the other $d-1$ dimensions. This is a result of w_i being defined as the product of weights in all dimensions, as seen in Eq. (5.3). The posterior mean of w_{im} is then $\langle p(w_{im} = 1|y_i, \mathbf{z}_i, \mathbf{x}_i, \boldsymbol{\theta}, w_{i,k\neq m})\rangle$, and:

$$\langle w_i \rangle = \prod_{m=1}^{d} \langle w_{im} \rangle$$

where $\langle . \rangle$ denotes the expectation operator.

The final posterior EM update equations (along with posterior updates for w_{im}) are listed in below:

E-step:

$$\boldsymbol{\Sigma}_{\mathbf{b}_m} = \left(\boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} + \sum_{i=1}^{N} \langle w_i \rangle \, \mathbf{x}_{im} \mathbf{x}_{im}^T \right)^{-1} \tag{5.13}$$

$$\langle \mathbf{b}_m \rangle = \boldsymbol{\Sigma}_{\mathbf{b}_m} \left(\sum_{i=1}^{N} \langle w_i \rangle \, \langle \mathbf{z}_{im} \rangle \, \mathbf{x}_{im} \right) \tag{5.14}$$

$$\boldsymbol{\Sigma}_{\mathbf{z}_i|y_i,\mathbf{x}_i} = \frac{\boldsymbol{\Psi}_{zN}}{\langle w_i \rangle} - \frac{1}{s_i} \left(\frac{\boldsymbol{\Psi}_{zN}}{\langle w_i \rangle} \mathbf{1}\mathbf{1}^T \frac{\boldsymbol{\Psi}_{zN}}{\langle w_i \rangle} \right) \tag{5.15}$$

$$\langle \mathbf{z}_i \rangle = \frac{\boldsymbol{\Psi}_{zN}\mathbf{1}}{s_i \langle w_i \rangle} + \left(\mathbf{I}_{d,d} - \frac{\boldsymbol{\Psi}_{zN}\mathbf{1}\mathbf{1}^T}{s_i \langle w_i \rangle} \right) \mathbf{bx_i} \tag{5.16}$$

$$\psi_{zmN} = \frac{\left(\begin{array}{c} \sum_{i=1}^{N} \langle w_i \rangle \left(\langle z_{im} \rangle - \langle \mathbf{b}_m \rangle^T \mathbf{x}_{im} \right)^2 + \sum_{i=1}^{N} \langle w_i \rangle \boldsymbol{\Sigma}_{\mathbf{z}_i | y_i, \mathbf{x}_i} \\ + n_{m0} \psi_{zmN0} + \langle \mathbf{b}_m \rangle^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \langle \mathbf{b}_m \rangle \end{array} \right)}{n_{m0} + \sum_{i=1}^{N} \langle w_i \rangle} \tag{5.17}$$

$$\langle w_{im} \rangle = \frac{q_{im} A_i^{\prod_{k=1,k\neq m}^{d} \langle w_{ik} \rangle}}{q_{im} A_i^{\prod_{k=1,k\neq m}^{d} \langle w_{ik} \rangle} + 1 - q_{im}} \tag{5.18}$$

$$\langle h_m \rangle = \frac{a_{hm0} + N - \sum_{i=1}^{N} \langle w_{im} \rangle}{b_{hm0} + \sum_{i=1}^{N} \lambda_{im} \left(x_{im} - x_{qm} \right)^{2r}} \tag{5.19}$$

M-step:

$$\sigma^2 = \frac{1}{\sum_{i=1}^{N} \langle w_i \rangle} \sum_{i=1}^{N} \left\langle w_i \left(y_i - \mathbf{1}^T \mathbf{z}_i \right)^2 \right\rangle \tag{5.20}$$

$$\lambda_{im} = \frac{1}{1 + \left(x_{im} - x_{qm} \right)^{2r} h_m} \tag{5.21}$$

where:

$$\begin{aligned} \langle w_i \rangle &= \prod_{m=1}^{d} \langle w_{im} \rangle \\ s_i &= \sigma^2 + \mathbf{1}^T \frac{\boldsymbol{\Psi}_{zN}}{\langle w_i \rangle} \mathbf{1} \\ q_{im} = \lambda_{im} &= \frac{1}{1 + \left(x_{im} - x_{qm} \right)^{2r} \langle h_m \rangle} \\ A_i &= \text{Normal} \left(y_i; \mathbf{1}^T \langle \mathbf{z}_i \rangle, \sigma^2 \right) \prod_{m=1}^{d} \text{Normal} \left(z_{im}; \langle \mathbf{b}_m \rangle^T \mathbf{x}_{im}, \psi_{zm} \right) \end{aligned}$$

and $\mathbf{I}_{d,d}$ is a $d \times d$ identity matrix, $\mathbf{bx_i}$ is a d by 1 vector with coefficients $\langle \mathbf{b}_m \rangle^T \mathbf{x}_{im}$, $\boldsymbol{\Psi}_{zN}$ is a diagonal matrix with ψ_{zN} on its diagonal, Note that to avoid division by zero, $\langle w_i \rangle$ needs to be capped to some small non-zero value. Please refer to Appendix D.2 for derivations of the EM update equations listed above.

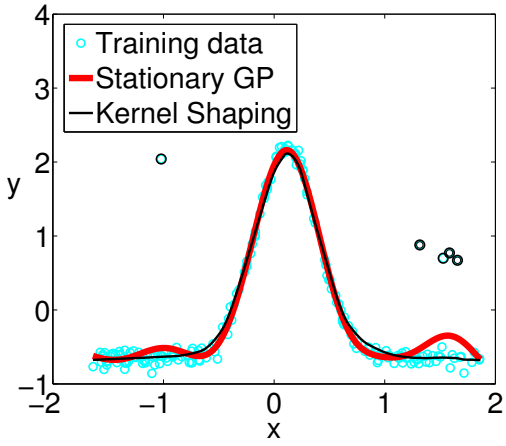


Figure 5.3: Effect of outliers (in black circles) on local kernel shaping

Closer examination of Eq. (5.14)—the expression for $\langle \mathbf{b}_m \rangle$—shows that it is a standard Bayesian weighted regression update (Gelman et al. 2000), i.e., a data sample i with lower weight w_i will be downweighted in the regression. Since the weights are influenced by the residual error at each data point, as Eq. (5.18) shows, an outlier will be downweighted appropriately and eliminated from the local model. Figure 5.3 shows how local kernel shaping is able to ignore outliers that a classical GP fits. Note that the posterior mean of h_m is a function of the number of samples that are excluded from the local model and the weighted distance that each sample is from the query point (i.e., the denominator of h_m).

Initialization of Priors: A few remarks should be made regarding the initialization of priors used in Eqs. (5.13) to (5.21). $\mathbf{\Sigma}_{\mathbf{b}_m,0}$ can be set to $10^6\mathbf{I}$ to reflect a large uncertainty associated with the prior distribution of $\mathbf{b}$. The initial noise variance, $\psi_{zm,0}$, should be set to the best guess on the noise variance. To adjust the strength of this prior, n_{m0} can

be set to the number of samples one believes to have seen with noise variance $\psi_{zm,0}$— the more points, the strong the prior. For instance, n_{m0} could be set to a fraction of the number of samples in the training set. Finally, the initial h of the weighting kernel should be set so that the kernel is broad and wide. We use values of $a_{hm0} = b_{hm0} = 10^{-6}$ so that $h_{m0} = 1$ with high uncertainty. In a similar fashion to outlier detection in Section 4, some sort of initial belief about the noise level is needed to distinguish between noise and structure in the training data. Aside from the initial prior on ψ_{zm}, we used the same priors for all data sets in our evaluations.

5.1.3 Computational Complexity

For one local model, the EM update equations have *a computational complexity of* $O(Nd)$ *per EM iteration*, where d is the number input dimensions and N is the size of the training set. This efficiency arises from the introduction of the hidden random variables $\mathbf{z}_i$, which allows $\langle \mathbf{z}_i \rangle$ and $\mathbf{\Sigma}_{\mathbf{z}_i|y_i,\mathbf{x}_i}$ to be computed in $O(d)$ and avoids a $d \times d$ matrix inversion which would typically require $O(d^3)$.

Nonstationary GP methods, e.g., (Paciorek & Schervish 2004), require $O(N^3) + O(N^2)$ for training and prediction, while other more efficient stationary GP methods, e.g., (Snelson & Ghahramani 2006*b*), have $O(M^2N) + O(M^2)$ training and prediction costs (where $M << N$ is the number of pseudoinputs used in Snelson & Ghahramani (2006*b*)).

Our algorithm requires $O(NdI_{EM})$, where I_{EM} is the number of EM iterations (with a maximal cap of 1000 iterations used). The algorithm also does not require any MCMC

sampling as in Rasmussen & Ghahramani (2002) and Meeds & Osindero (2005), making it more appealing to real-time applications.

5.2 Extension to Gaussian Processes

We can apply the algorithm in Section 5.1 not only to locally weighted learning with linear models, but also to derive a non-stationary GP method. A GP is defined by a mean and and a covariance function, where the covariance function K captures dependencies between any two points as a function of the corresponding inputs, i.e., $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \text{cov}\left(f(\mathbf{x}_i), f(\mathbf{x}'_j)\right)$, where $i, j = 1, .., N$. Standard GP models use a stationary covariance function, where the covariance between any two points in the training data is a function of the distances $|\mathbf{x}_i - \mathbf{x}_j|$, not on their locations. Stationary GPs perform suboptimally for functions that have different properties in various parts of the input space (e.g., discontinuous functions) where the stationary assumption fails to hold.

Various methods have been proposed to specify non-stationary GPs. These include defining a non-stationary Matérn covariance function (Paciorek & Schervish 2004), adopting a mixture of local experts approach (Tresp 2000, Rasmussen & Ghahramani 2002, Shi, Murray-Smith & Titterington 2005, Meeds & Osindero 2005) to use independent GPs to cover data in different regions of the input space, and using multidimensional scaling to map a non-stationary spatial GP into a latent space (Schmidt & O'Hagan 2003).

Given the data set D drawn from the function $y = f(\mathbf{x}) + \epsilon$, as previously introduced in Section 5.1.1, we propose an approach to specify a non-stationary covariance function.

Assuming the use of a quadratic negative exponential covariance function, the covariance function of a stationary GP is as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j) = v_1^2 \exp(-0.5 \sum_{m=1}^{d} h_m (x_{im} - x'_{jm})^2) + v_0$$

where the $(d+2)$ hyperparameters $\{h_1, h_2, ..., h_d, v_0, v_1\}$ are optimized. In a non-stationary GP, the covariance function could then take the following form (Gibbs 1997):

$$k(\mathbf{x}_i, \mathbf{x}_j) = v_1^2 \left(\frac{2\sqrt{h_{im} h_{jm}}}{h_{im} + h_{jm}} \right)^{1/2} \exp\left(- \sum_{m=1}^{d} \frac{(x_{im} - x_{jm})^2}{2(h_{im} + h_{jm})} \right) + v_0 \qquad (5.22)$$

where h_{im} is the bandwidth of the local model centered at x_{im} and h_{jm} is the bandwidth of the local model centered at x_{jm}. The hyperparameters to be optimized now consist of only v_0 and v_1.

We learn first the values of $\{h_{im}\}_{m=1}^{d}$ for all training data samples $i = 1, ..., N$, using our proposed local kernel shaping algorithm and then optimizing the hyperparameters v_0 and v_1. To make a prediction for a test sample $\mathbf{x}_q$, we learn also the values of $\{h_{qm}\}_{m=1}^{d}$, i.e., the bandwidth of the local model centered at $\mathbf{x}_q$. Importantly, since the covariance function of the GP is derived from locally constant models, we learn with locally constant, instead of locally linear, polynomials. We use $r = 1$ for the weighting kernel in order keep the degree of nonlinearity consistent with that in the covariance function (i.e., quadratic). Even though the weighting kernel used in the local kernel shaping algorithm is not a quadratic negative exponential, it has a similar bell shape, but with a flatter top and shorter tails. Because of this, our proposed method is an approximated form of

a non-stationary GP. Nonetheless, the augmented GP is able to capture non-stationary properties of the function f without needing MCMC sampling, unlike previously proposed non-stationary GP methods (Rasmussen & Ghahramani 2002, Meeds & Osindero 2005).

5.3 Experimental Results

We evaluate our Bayesian local kernel shaping algorithm on synthetic data sets, in order to establish that its performance is competitive to other state-of-the-art techniques for nonlinear regression such as locally weighted projection regression (LWPR) and Gaussian process regression (GPR). We also demonstrate the effectiveness of our algorithm on a real robotic data set, learning an inverse kinematics model for a robot arm.

5.3.1 Synthetic Data

First, we show our local kernel shaping algorithm's bandwidth adaptation abilities on several synthetic data sets, comparing it to a stationary GP and our proposed augmented non-stationary GP.

One-dimensional synthetic data: For the ease of visualization, we consider the following one-dimensional functions, similar to those in Paciorek & Schervish (2004):

(i) a function with a discontinuity

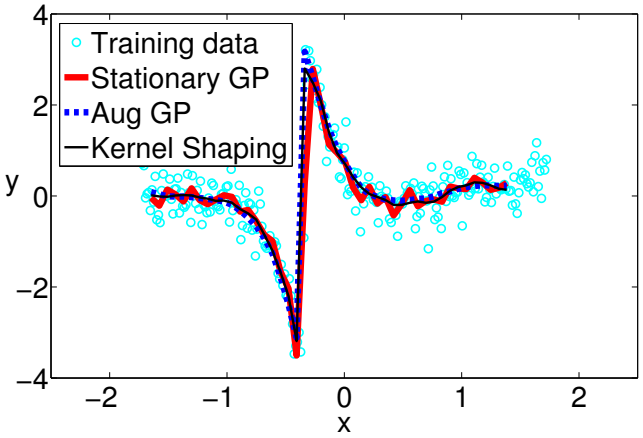
(ii) a spatially inhomogeneous function

(iii) a straight line

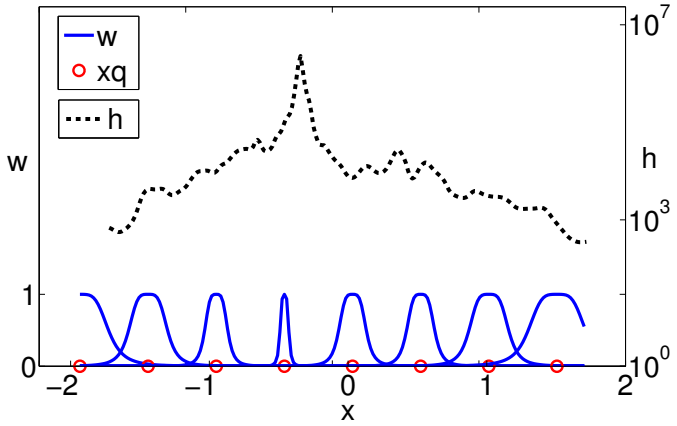
The data set for function (i) consists of 250 training samples, 201 test inputs (evenly spaced across the input space) and output noise with $\sigma^2 = 0.3025$. The data set for function ii) consists of 250 training samples, 101 test inputs and an output signal-to-noise ratio (SNR) of 10. The data set for function iii) has 50 training samples, 21 test inputs and an output SNR of 100.

Figures 5.4 to 5.6 show the predicted outputs of a stationary GP, augmented non-stationary GP and the local kernel shaping algorithm for data sets (i)-(iii). The local kernel shaping algorithm smoothes over regions where a stationary GP overfits and yet, it still manages to capture regions of highly varying curvature, as seen in Figures 5.4(a) and 5.6(a). It correctly adjusts the bandwidths h with the curvature of the function. When the data looks linear, the algorithm opens up the weighting kernel so that all data samples are considered, as Figure 5.6(b) shows. Our proposed augmented non-stationary GP also can handle the non-stationary nature of the data sets as well, and its performance is quantified in Table 5.1. Returning to our motivation to use these algorithms to obtain linearizations for learning control, it is important to realize that the high variations from fitting noise, as shown by the stationary GP in Figures 5.4(a) and 5.5(a), are detrimental for learning algorithms, as the slope (or tangent hyperplane, for high-dimensional data) would be wrong.

Table 5.1 reports the normalized mean squared prediction error (nMSE) values for function (i) and function (ii) data sets, averaged over 20 random data sets. We also performed Gibbs sampling Adaptive Rejection Sampling to sample from the conditional distribution of $\mathbf{h}$.

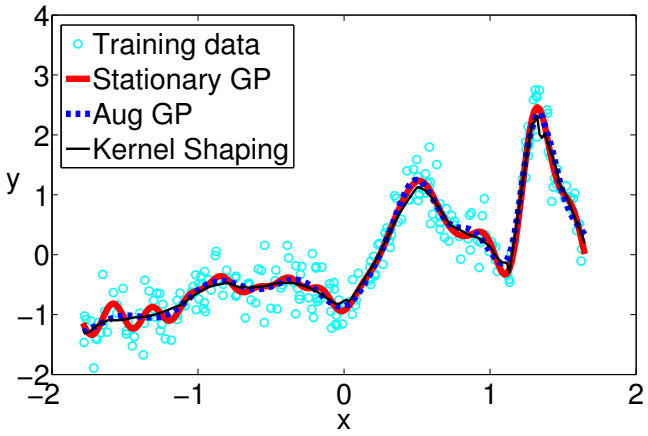


(a) Observed training data vs. predicted output for function (i)

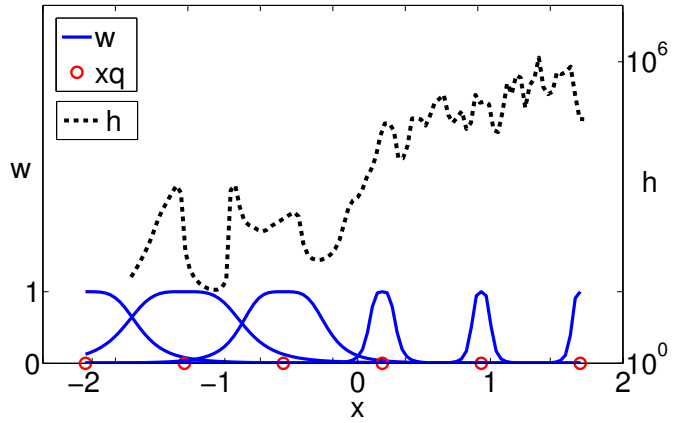


(b) Learnt weight and bandwidth from Bayesian local kernel shaping

Figure 5.4: Observed vs. predicted output made by stationary GP, augmented GP and Bayesian local kernel shaping for function (i). Figure 5.4(b) shows the bandwidths learnt by local kernel shaping and the corresponding weighting kernels (in dotted black lines) for various input query points (shown in red circles).

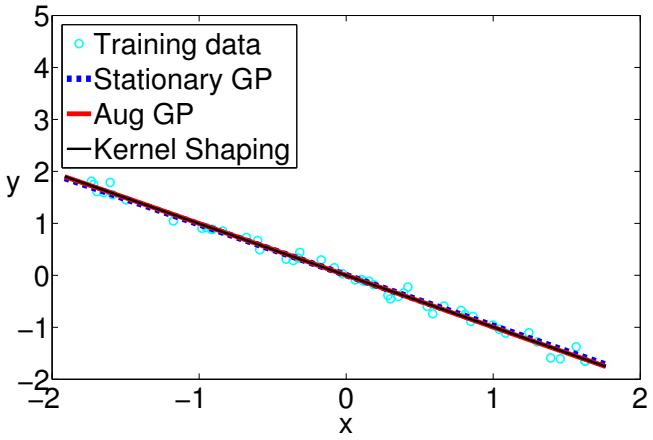


(a) Observed training data vs. predicted output for function (ii)

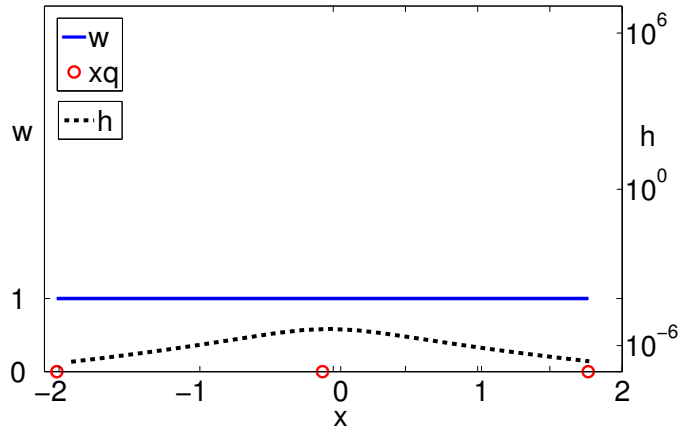


(b) Learnt weight and bandwidth from Bayesian local kernel shaping

Figure 5.5: Observed vs. predicted output made by stationary GP, augmented GP and Bayesian local kernel shaping for function (ii). Figure 5.5(b) shows the bandwidths learnt by local kernel shaping and the corresponding weighting kernels (in dotted black lines) for various input query points (shown in red circles).



(a) Observed training data vs. predicted output for function (iii)



(b) Learnt weight and bandwidth from Bayesian local kernel shaping

Figure 5.6: Observed vs. predicted output made by stationary GP, augmented GP and Bayesian local kernel shaping for function (iii). Figure 5.6(b) shows the bandwidths learnt by local kernel shaping and the corresponding weighting kernels (in dotted black lines) for various input query points (shown in red circles).

Table 5.1: Average normalized mean squared prediction error values, for a stationary GP model, our augmented non-stationary GP, local kernel shaping. Results, averaged over 20 random data sets, are shown for functions (i) and (ii).

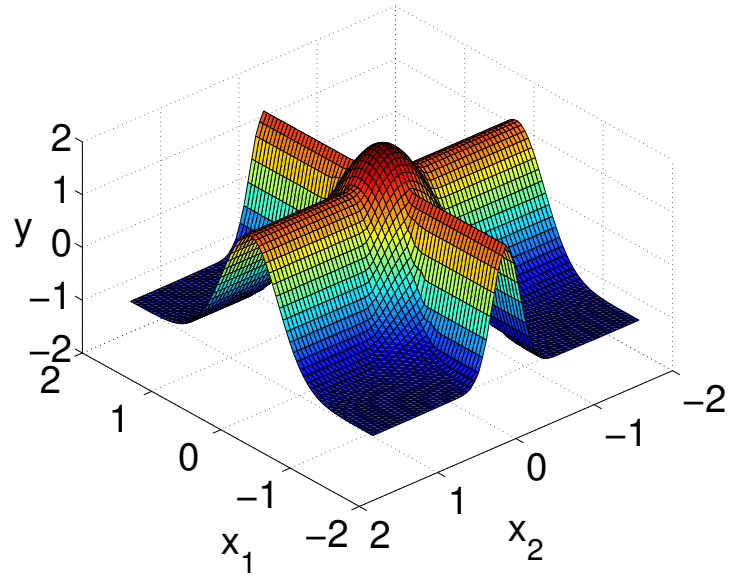
Method	Function (i)	Function (ii)
Stationary GP	0.1251 ± 0.013	0.0230 ± 0.0047
Augmented non-stationary GP	0.0110 ± 0.0078	0.0212 ± 0.0067
Local Kernel Shaping	0.0092 ± 0.0068	0.0217 ± 0.0058

2-dimensional synthetic data: We also evaluated Bayesian local kernel shaping on 500 noisy training data samples, drawn from the 2-dimensional function (CROSS 2D), generated from:

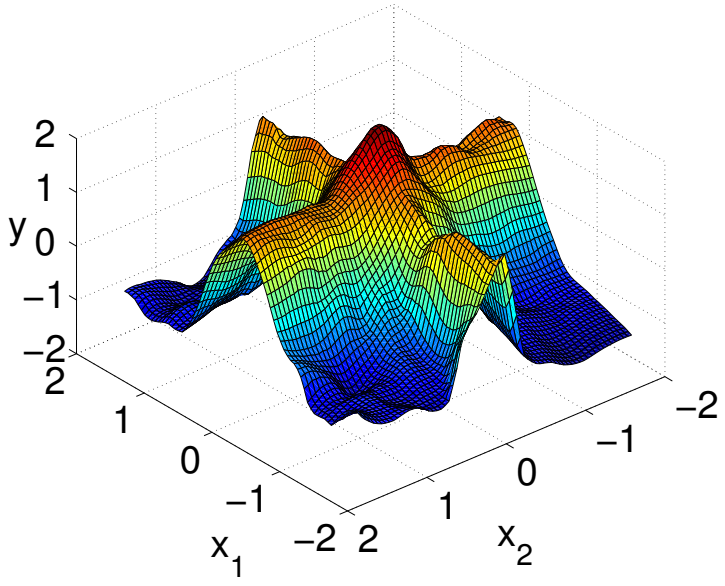
$$y = \max\left\{\exp(-10x_1^2), \exp(-50x_2^2), 1.25\exp(-5(x_1^2 + x_2^2))\right\}$$

as previously examined in Schaal et al. (1998) and Vijayakumar et al. (2005). Mean-zero noise with a variance of 0.01 was added to the outputs. Figure 5.7(a) shows the target function, evaluated over a noiseless test input, resulting in 1681 data points on a 41×41 grid in the 2×2 square in input space. Figure 5.7(b) shows the predicted outputs for Bayesian local kernel shaping, and Figure 5.8 depicts the learnt distance metric values h over a subset of the test data points scattered over the 41×41 grid (shown as red circles). As before, we see that the width of the weighting kernel adjusts according to the curvature of the function.

Table 5.2 compares the performance of Bayesian local kernel shaping to GPR and LWPR, averaged over 10 randomly chosen training data sets. Performance was quantified in terms of normalized mean squared prediction error (nMSE) value on the noiseless test sets.



(a) Target Function



(b) Predicted Function by Bayesian local kernel shaping

Figure 5.7: a) nonlinear 2-dimensional CROSS target function to be approximated; b) approximated/predicted function produced by Bayesian local kernel shaping

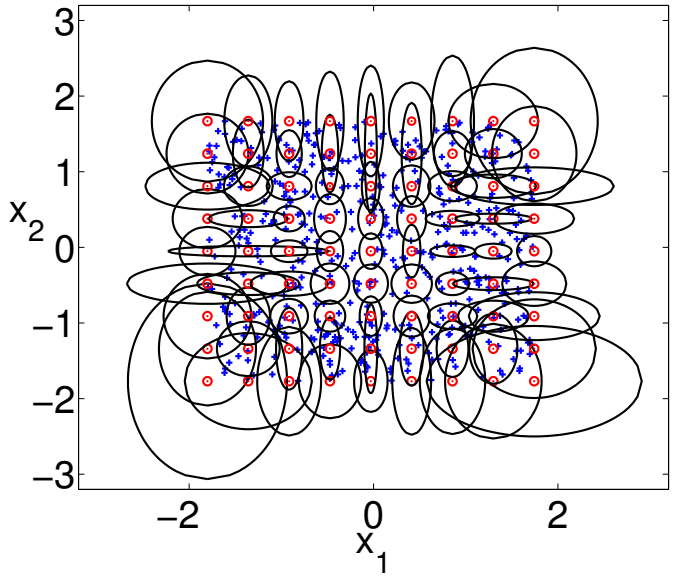


Figure 5.8: Learnt weighting kernels in input space for 2-dimensional CROSS target function in Figure 5.7. Small red circles indicate the test input points and centers of the weighting kernels. Training data has an output noise has a variance of 0.01.

Table 5.2: Average normalized mean squared error (nMSE) comparisons between Gaussian Process regression (GPR), LWPR and Bayesian local kernel shaping for the nonlinear 2-d CROSS function. Results shown are averaged over 10 trials and the training data set has $N = 500$ samples.

Algorithm	nMSE	std-dev
GPR	0.01991	0.00314
LWPR	0.02556	0.00416
Local Kernel Shaping	0.02609	0.00532

Motorcycle data set: Figures 5.9 and 5.10 show the results of the local kernel shaping algorithm and the proposed augmented non-stationary GP on the "real-world" motorcycle data set (Silverman 1985) consisting of 133 samples (with 80 equally spaced input query points used for prediction). We also show results from two previously proposed MCMC-based non-stationary GP methods: an infinite mixture of GP experts from Rasmussen & Ghahramani (2002), shown in Figure 5.11, and an alternate infinite mixture of GP experts (Meeds & Osindero 2005), shown in Figure 5.12. We can see that the augmented non-stationary GP and the local kernel shaping algorithm both capture the leftmost flatter region of the function, as well as some of the more nonlinear and noisier regions after 30msec.

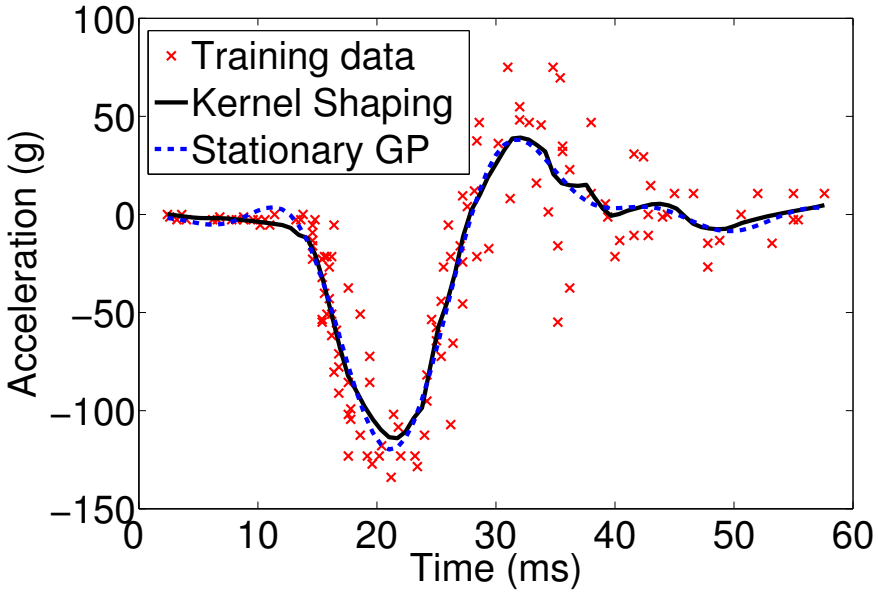


Figure 5.9: Predicted results from Bayesian local kernel shaping on the motorcycle impact data set from Silverman (1985),

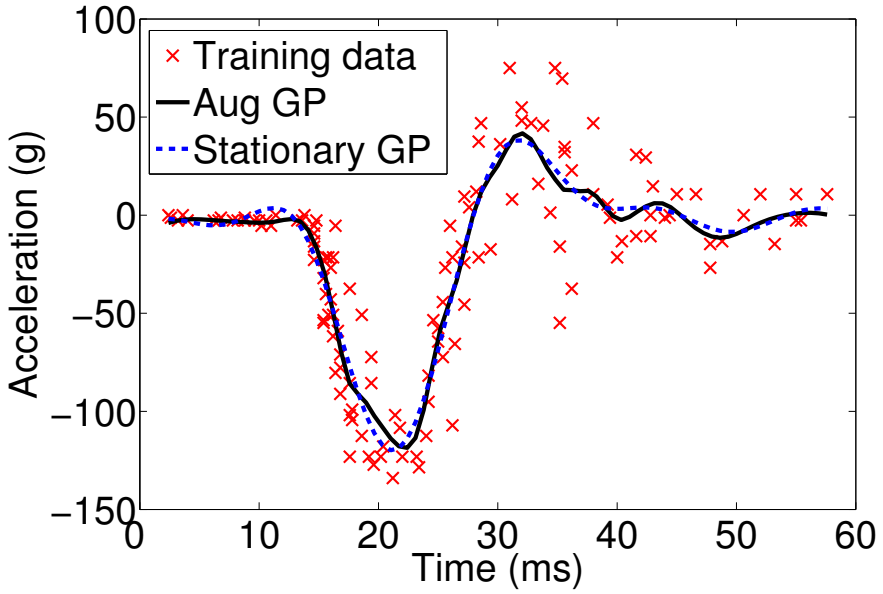


Figure 5.10: Predicted results from our augmented GP method on the motorcycle impact data set from Silverman (1985),

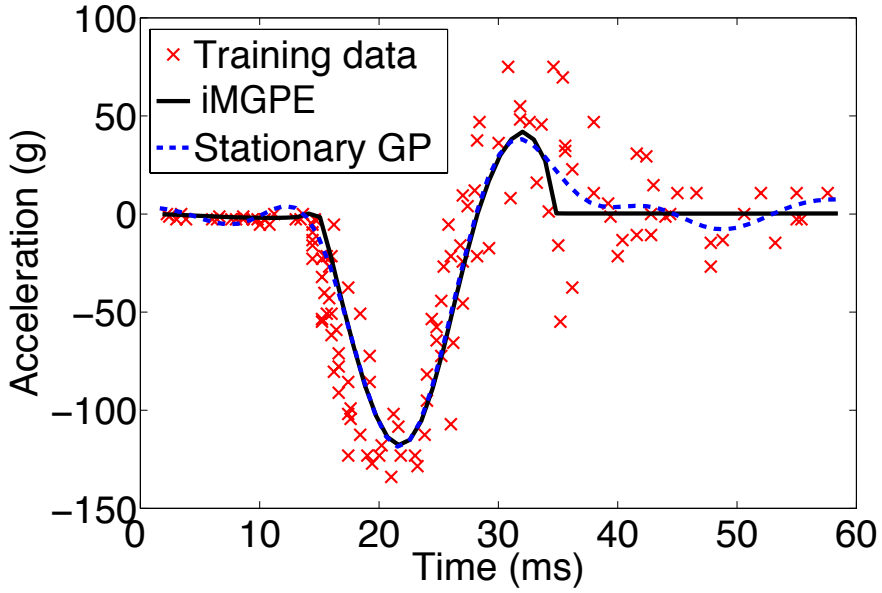


Figure 5.11: Predicted results from infinite mixture of GP experts (iMGPE) on the motorcycle impact data set from Silverman (1985), Graph is taken from Rasmussen & Ghahramani (2002).

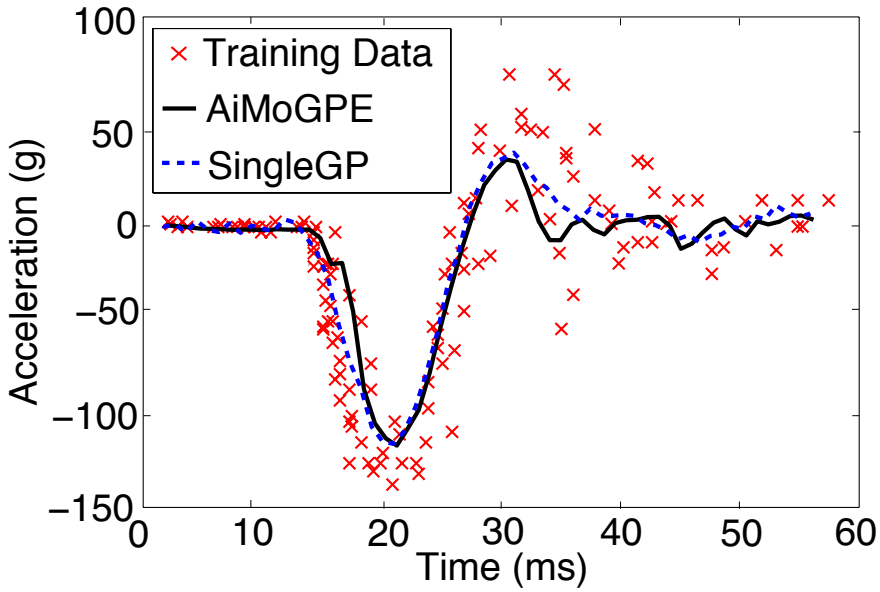


Figure 5.12: Predicted results from alternate infinite mixture of GP experts (AiMoGPE) on the motorcycle impact data set from Silverman (1985), Graph is taken from Meeds & Osindero (2005).

Figure 5.13: SensAble Phantom haptic robotic arm

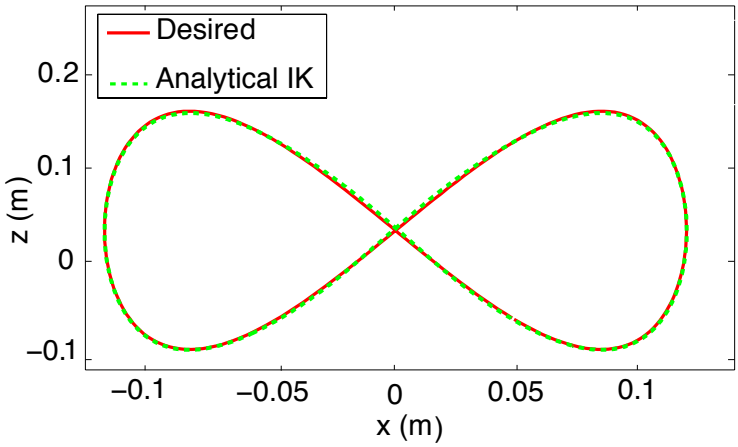
5.3.2 Robot Data

Next, we move on to an example application: learning an inverse kinematics model for a 3 degree-of-freedom (DOF) haptic robot arm (manufactured by SensAble, shown in Figure 5.13) in order to control the end-effector along a desired trajectory. This will allow us to verify that the kernel shaping algorithm can successfully deal with a large, noisy real-world data set with outliers and non-stationary properties—typical characteristics of most control learning problems.

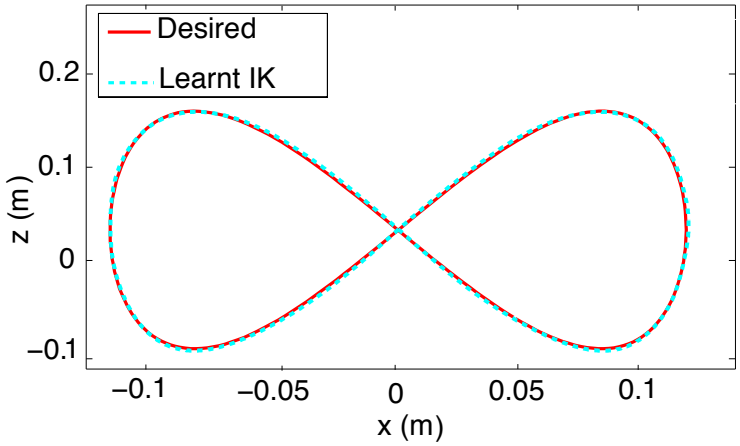
We collected $60,000$ data samples from the arm while it performed random sinusoidal movements within a constrained box volume of Cartesian space. Each sample consists of the arm's joint angles $\mathbf{q}$, joint velocities $\dot{\mathbf{q}}$, end-effector position in Cartesian space $\mathbf{x}$, and end-effector velocities $\dot{\mathbf{x}}$. From this data, we first learn a forward kinematics model:

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{5.23}$$

where $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix. The transformation from $\dot{\mathbf{q}}$ to $\dot{\mathbf{x}}$ can be assumed to be locally linear at a particular configuration $\mathbf{q}$ of the robot arm. We learn the forward model using kernel shaping, building a local model around each training point only if that point is not already sufficiently covered by an existing local model (e.g., having an activation weight of less than 0.2). Using insights into robot geometry, we localize the models only with respect to $\mathbf{q}$ while the regression of each model is trained only on a mapping from $\dot{\mathbf{q}}$ to $\dot{\mathbf{x}}$—these geometric insights are easily incorporated as priors in the Bayesian model. This procedure resulted in 56 models being built to cover the entire space of training data.



(a) Analytical solution



(b) Learnt solution

Figure 5.14: Desired versus actual trajectories for SensAble Phantom robot arm

We artificially introduce a redundancy in our inverse kinematics problem on the 3-DOF arm by specifying the desired trajectory $(\mathbf{x}, \dot{\mathbf{x}})$ only in terms of x, z positions and velocities, i.e., the movement is supposed to be in a vertical plane in front of the robot. Analytically, the inverse kinematics equation is:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}(\mathbf{q})\dot{\mathbf{x}} - \alpha(\mathbf{I} - \mathbf{J}^{\#}\mathbf{J})\frac{\partial g}{\partial \mathbf{q}} \tag{5.24}$$

where $J^{\#}(\mathbf{q})$ is the pseudo-inverse of the Jacobian. The second term is an optimal solution to the redundancy problem, specified here by a cost function g in terms of joint angles $\mathbf{q}$.

To learn a model for $\mathbf{J}^{\#}$, we can reuse the local regions of $\mathbf{q}$ from the forward model, where $\mathbf{J}^{\#}$ is also locally linear. The redundancy issue can be solved by applying an additional weight to each data point according to a reward function (Peters & Schaal 2008). In our case, the task is specified in terms of $\{\dot{x}, \dot{z}\}$, so we define a reward based on a desired y coordinate, y_{des}, that we would like to enforce as a soft constraint. Our reward function is $g = e^{-\frac{1}{2}h(k(y_{des}-y)-\dot{y})^2}$, where k is a gain and h specifies the steepness of the reward. This ensures that the learnt inverse model chooses a solution which produces a $\dot{y}$ that pushes the y coordinate toward y_{des}. We invert each forward local model using a weighted linear regression, where each data point is weighted by the weight from the forward model and additionally weighted by the reward.

We test the performance of this inverse model (Learnt IK) in a figure-eight tracking task as shown in Figure 5.14. As seen, the learnt model performs as well as the analytical inverse kinematics solution (IK), with root mean squared tracking errors in positions and

velocities very close to that of the analytical solution. This demonstrates that kernel shaping is an effective learning algorithm for use in robot control learning applications.

Applying any arbitrary nonlinear regression method (such as a GP) to the inverse kinematics problem would, in fact, lead to unpredictably bad performance. The inverse kinematics problem is a one-to-many mapping and requires careful design of a learning problem to avoid problems with non-convex solution spaces (Jordan & Rumelhart 1991). Our suggested method of learning linearizations with a forward mapping (which is a proper function), followed by learning an inverse mapping within the local region of the forward mapping, is one of the few clean approaches to the problem. Instead of using locally linear methods, one could also use density-based estimation techniques like mixture models (Ghahramani 1994). However, these methods must select the correct mode in order to arrive at a valid solution, and this final step may be computationally intensive or involve heuristics. For these reasons, applying a MCMC-type approach or GP-based method to the inverse kinematics problem was omitted as a comparison.

5.4 Discussion

We introduced a Bayesian formulation of spatially local adaptive kernels for locally weighted regression. The local kernel shaping algorithm is computationally efficient, making it suitable for large data sets, and can handle non-stationary functions where the data density, curvature and amount of noise in the data vary spatially. The algorithm can also be integrated into nonlinear algorithms such as GPs, offering an approach that does not require any sampling and making it more appealing for real-time problems.

We show that our local kernel shaping method is particularly useful for learning control, demonstrating results on an inverse kinematics problem, and envision extensions to more complex problems with redundancy, e.g., learning inverse dynamics models of complete humanoid robots.

Note that our algorithm requires only one prior be set by the user, i.e., the prior on the output noise. All other biases are initialized the same for all data sets and kept uninformative. In general, the bias on the output noise is needed in order to distinguish structure from noise in data—which is also the case for outlier detection. In our algorithm, the noise bias needs to be specified in a very coarse way.

In its current form, our Bayesian kernel shaping algorithm is built for high-dimensional inputs due to its low computational complexity: it scales linearly with the number of input dimensions. However, numerical problems may arise in case of redundant and irrelevant input dimensions. Future work will address this issue through the use of an automatic relevant determination feature. Note that all comments about the functionality of local function approximation in high-dimensional spaces are the same as those made in (Vijayakumar et al. 2005). Other future extensions include an online implementation of the local kernel shaping algorithm.

Chapter 6

Conclusion

In this dissertation, we presented a series of Bayesian algorithms for the development of autonomous learning systems. The algorithms are black-box-like in property, requiring minimal tuning of parameters by the user. A Bayesian approach allows the user to express domain-specific knowledge intuitively. Instead of having to search for specific values or range of values to set parameters to, the user need only to specify *a priori* probability distribution over parameters.

6.1 Summary of Dissertation Contributions

Algorithmic contributions:

- Reformulation of linear regression as a sparse, numerically robust, computationally efficient (for an iterative method), automatic and truly black-box method
- Accurate model identification (of a generative model) with input and output noise
- Automatic outlier detection
- Nonlinear regression for heteroscedastic and non-stationary functions

Contributions to application domains:

- Modeling of data for brain-machine interfaces
- Accurate parameter identification for rigid body dynamics
- Robust state estimation for tracking
- Learning nonlinear internal models for robotics

6.2 Summary of Chapters

6.2.1 Chapter 2

Chapter 2 addresses how linear regression in high dimensions can be performed, especially where the input data has a large number of input dimensions—many of which are redundant or irrelevant. These kinds of data are typically encountered in the fields of neuroscience and brain-machine interfaces.

Variational Bayesian least squares, the algorithm introduced in the chapter, is computationally efficient, requiring $O(d)$ updates per EM iteration—where d is the input dimensionality, is guaranteed to converge to the global solution and has no parameters to be tuned. The initial prior distribution over the precision variables α can be set to be wide and uninformative. The hyperparameter values for the initial prior of α need never be changed from data set to data set, making VBLS autonomous and "black-box"-like.

Even though VBLS is an iterative statistical method, it can be embedded into other iterative methods to make them more computationally efficient. Its iterative nature also makes it suitable for real-time learning scenarios, where time constraints dictate that an

approximately accurate solution is better than an extremely accurate one that takes too long to compute.

6.2.2 Chapter 3

Chapter 3 considers high-dimensional scenarios introduced in Chapter 2, but takes into account the more realistic scenario where observed, sensory data (in particular, observed input data) is contaminated with noise. A novel full Bayesian treatment of the linear system identification is introduced in the chapter. The algorithm is robust to high-dimensional, ill-conditioned data with noisy input and output data, remains computationally efficient (again, $O(d)$ per EM iteration) and has no parameters that need to be tuned.

The algorithm is, however, iterative, but then again, so is probabilistic factor analysis for regression. As with VBLS, the iterative nature of the algorithm makes it suitable for real-time learning scenarios and also for incorporation into other iterative methods to de-noise input data.

The algorithm was applied to the problem of estimating parameters in rigid body dynamics—an estimation problem that is linear in the unknown parameters. Physical constraints of the parameters (such as positive mass, a positive definite inertial matrix, the parallel axis theorem, etc.) had to be satisfied. As a result, we introduced an additional post-processing step after the algorithm is executed in order to ensure resulting parameter estimates satisfy physical constraints. The additional post-processing step projects the inferred solution onto the feasible space of possible solutions.

The chapter focuses on the problem of learning for system identification (for the purpose of finding the true parameters of a system), versus learning for prediction. Good prediction is often possible without modeling all components of the generative system, while parameter estimation comes useful when the identified model is used in other ways that are different than in the training scenario. For example, in the application domain of robotics, the analytical inverse of the identified model is often needed in model-based control.

6.2.3 Chapter 4

Chapter 4 examines how outliers in noisy, sensory data can be detected and removed, especially in real-time. In particular, we propose an automatic approach to outlier detection and removal that can be incorporated into the Kalman filter, as well as other more complex filters.

The Bayesian weighted regression framework introduced in the chapter is easy to use. It does not require any parameter tuning beyond the weight prior (which is, in fact, an outlier prior, needed in order to distinguish outliers from data structure), interference from the user, heuristics or sampling. In this framework, each data sample has a scalar weight associated with it and the optimal value of the weights are learnt.

6.2.4 Chapter 5

Finally, chapter 5 moves on to the nonlinear high-dimensional regression problem. We introduce a Bayesian kernel shaping algorithm that automatically learns the bandwidth of a local model. The Bayesian kernel shaping algorithm is computationally efficient and

can handle non-stationary functions where the data density, curvature and amount of noise in the data vary spatially. The algorithm can be leveraged in not only in locally methods, such as locally weighted regression, but also in global methods that are linear in the parameters, such as Gaussian process regression, in order to capture non-stationary properties in data.

The algorithm requires that only one prior be set by the user, i.e., the output noise prior. Similar to outlier detection, this noise prior is needed to distinguish noise from structure in the data. The noise bias in Bayesian kernel shaping needs to be specified coarsely.

6.3 Opportunities for Future Research

The next step towards realizing autonomous systems is to bring the Bayesian kernel shaping algorithm developed in Chapter 5, as well as the other algorithms, to real-time learning scenarios. The version of Bayesian kernel shaping presented in this thesis needs also be modified to handle data with many redundant and irrelevant input features. A potential extension is to apply the estimation of a local model's bandwidth from a spatial to a temporal setting—specifically, to the estimation of forgetting rates in incremental learning (e.g., as in Recursive Least squares). Estimating the optimal value of forgetting rates is equivalent to determining the window of previously observed samples to consider and average over when trying to make predictions.

A related, interesting research direction is the problem of online model selection and how it can be done in an autonomous, black-box-like manner. The order in which samples are observed will affect the resulting model that is selected. Nevertheless, if one could determine whether a sample would be discarded or taken into account once it is observed, then the resulting mechanism may be effective alternative solution to bandwidth adaptation.

Reference List

Abe, N., Zadrozny, B. & Langford, J. (2006), Outlier detection by active learning, *in* 'Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM Press, New York, NY, USA, pp. 767–772.

Aitkin, M. & Wilson, G. T. (1980), 'Mixture models, outliers and the EM algorithm', *Technometrics* **22**, 325–331.

Akaike, H. (1974), 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control* **19**, 716–723.

An, C. H., Atkeson, C. G. & Hollerbach, J. M. (1988), *Model-based control of a robot manipulator*, MIT Press.

Atkeson, C., Moore, A. & Schaal, S. (1997), 'Locally weighted learning', *AI Review* **11**, 11–73.

Atkeson, C. & Schaal, S. (1997), Robot learning from demonstration, *in* 'Proceedings of the 14th international conference on Machine learning', Morgan Kaufmann, pp. 12–20.

Attias, H. (2000), A Variational Bayesian framework for graphical models, *in* 'Advances in Neural Information Processing Systems 13', MIT Press.

Azzalini, A. & Bowman, A. W. (1990), 'A look at some data on the Old Faithful geyser', *Applied Statistics* **39**, 357–365.

Bar-Shalom, Y., Li, X. R. & Kirubarajan, T. (2001), *Estimation with Applications to Tracking and Navigation*, Wiley.

Beal, M. J. (2003), Variational algorithms for approximate Bayesian inference, PhD thesis, Gatsby Computational Neuroscience Unit, University College London.

Belsley, D. A., Kuh, E. & Welsch, R. E. (1980), *Regression diagnostics: Identifying influential data and sources of collinearity*, Wiley.

Bernardo, J. M. & Smith, A. F. (1994), *Bayesian Theory*, John Wiley.

Bierman, G. J. (1977), 'Factorization methods for discrete sequential estimation', *Mathematics in Science and Engineering* **128**.

Bishop, C. M. (2006), *Pattern recognition and Machine learning*, Springer.

Breitenbach, M. & Grudic, G. Z. (2005), Clustering through ranking on manifolds, *in* 'Proceedings of the 22nd International Conference on Machine Learning', ACM Press, New York, NY, USA, pp. 73–80.

Chan, S. C., Zhang, Z. G. & Tse, K. W. (2005), A new robust Kalman filter algorithm under outliers and system uncertainties, *in* 'Proceedings of the IEEE International Symposium on Circuits and Systems', IEEE Press, pp. 4317–4320.

Chow, E. & Saad, Y. (1998), 'Approximate inverse preconditioners via sparse-sparse iterations', *SIAM Journal of Scientific Computing* **19**(3), 995–1023.

Csato, L. & Opper, M. (2002), 'Sparse online Gaussian processes', *Neural Computation* **14**(2), 641–669.

Dempster, A., Laird, N. & Rubin, D. (1977), 'Maximum likelihood from incomplete data via the EM algorithm', *Journal of Royal Statistical Society. Series B* **39**(1), 1–38.

Derksen, S. & Keselman, H. (1992), 'Backward, forward and stepwise automated subset selection algorithms: frequency of obtaining authentic and noise variables', *British Journal of Mathematical and Statistical Psychology* **45**, 265–282.

Draper, N. R. & Smith, H. (1981), *Applied Regression Analysis*, Wiley.

D'Souza, A. (2004), Towards tractable parameter-free statistical learning, PhD thesis, Department of Computer Science, University of Southern California.

D'Souza, A., Vijayakumar, S. & Schaal, S. (2004), The Bayesian backfitting relevance vector machine, *in* 'Proceedings of the 21st International Conference on Machine Learning', ACM Press.

Durovic, Z. M. & Kovacevic, B. D. (1999), 'Robust estimation with unknown noise statistics', *IEEE Transactions on Automatic Control* **44**, 1292–1296.

Fan, J. & Gijbels, I. (1992), 'Variable bandwidth and local linear regression smoothers', *Annals of Statistics* **20**, 2008–2036.

Fan, J. & Gijbels, I. (1995), 'Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation', *Journal of the Royal Statistical Society B* **57**, 371–395.

Fan, J. & Gijbels, I. (1996), *Local polynomial modeling and its applications*, Chapman and Hall.

Faul, A. C. & Tipping, M. E. (2001), A variational approach to robust regression., *in* 'Proceedings of the International Conference on Artificial Neural Networks', pp. 95–102.

Figueiredo, M. (2003), 'Adaptive sparseness for supervised learning', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**, 1150–1159.

Finnof, W., Hergert, F. & Zimmerman, H. G. (1993), 'Improving model selection by nonconvergent methods', *Neural Networks* **6**, 771–783.

Fischler, M. A. & Bolles, R. C. (1981), 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM* **24**(6), 381–395.

Fox, D., Burgard, W., Dellaert, D. & Thrun, S. (1999), Monte Carlo localization: efficient position estimation for mobile robots, *in* 'Proceedings of National Conference of Artificial Intelligence', pp. 343–349.

Frank, I. & Friedman, J. (1993), 'A statistical view of some chemometric regression tools', *Technometrics* **35**, 109–135.

Friedman, J. H. (1984), A variable span smoother, Technical report, Stanford University.

Friedman, J., Hastie, T. & Tibshirani, R. (2007), Pathwise coordinate optimization, Technical report, Department of Statistics, Stanford University.

Gelfand, A. E. (1996), Model determination using sampling-based methods, *in* W. R. Gilks, S. Richardson & D. J. Spiegelhalter, eds, 'Markov Chain Monte Carlo in Practice', Chapman and Hall, chapter 9, pp. 145–161.

Gelman, A., Carlin, J., Stern, H. & Rubin, D. (2000), *Bayesian Data Analysis*, Chapman and Hall.

Geman, S. & Geman, D. (1984), 'Stochastic relaxation, Gibbs distribution and Bayesian restoration of images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**, 721–741.

Ghahramani, Z. (1994), Solving inverse problems using an EM approach to density estimation, *in* 'Proceedings of the 1993 Connectionist Models summer school', Erlbaum Associates, pp. 316–323.

Ghahramani, Z. & Beal, M. (2000), Graphical models and variational methods, *in* D. Saad & M. Opper, eds, 'Advanced Mean Field Methods - Theory and Practice', MIT Press.

Ghahramani, Z. & Hinton, G. (1996), Parameter estimation for linear dynamical systems, Technical report, University of Toronto.

Ghahramani, Z. & Hinton, G. E. (1997), The EM algorithm for mixtures of factor analyzers, Technical report, University of Toronto.

Gibbs, M. N. (1997), Bayesian Gaussian Processes for Regression and Classification, PhD thesis, Department of Physics, University of Cambridge.

Golub, G. H. & Van Loan, C. (1989), *Matrix Computations*, John Hopkins University Press.

Goodman, J. (2004), Exponential priors for maximum entropy models, *in* 'Proceedings of the Annual Meeting of the Association for Computational Linguistics', Morgan Kaufmann.

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized additive models*, number 43 *in* 'Monographs on Statistics and Applied Probability', Chapman and Hall.

Hastings, W. K. (1970), 'Monte Carlo sampling methods using Markov chains and their applications', *Biometrika* **57**(1), 97–109.

Haynes, J. & Rees, G. (2005), 'Predicting the orientation of invisible stimuli from activity in human primary visual cortex', *Nature Neuroscience* **8**, 686.

Hoaglin, D. C. (1983), Letter values: a set of selected order statistics, *in* D. C. Hoaglin, F. Mosteller & J. W. Tukey, eds, 'Understanding Robust and Exploratory Data Analysis', Wiley, pp. 33–57.

Hoerl, A. & Kennard, R. W. (1970), 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics* **12**(3), 55–67.

Hollerbach, J. M. & Wampler, C. W. (1996), The calibration index and the role of input noise in robot calibration, *in* G. Giralt & G. Hirzinger, eds, 'Robotics Research: The Seventh International Symposium', Springer, pp. 558–568.

Huber, P. J. (1964), 'Robust estimation of a location parameter', *Annals of Mathematical Statistics* **35**, 73–101.

Huber, P. J. (1973), *Robust Statistics*, Wiley.

Hubert, M., Rousseeuw, P. & Vanden Branden, K. (2005), 'Robpca: a new approach to robust principal component analysis', *Technometrics* **47**, 64–79.

Jaakkola, T. S. & Jordan, M. I. (2000), 'Bayesian parameter estimation via variational methods', *Statistics and Computing* **10**, 25–37.

Jazwinski, A. H. (1970), *Stochastic Processes and Filtering Theory*, Academic Press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. & Saul, L. K. (1999), An introduction to variational methods for graphical models, *in* M. I. Jordan, ed., 'Learning in Graphical Models', MIT Press.

Jordan, M. I. & Rumelhart, D. E. (1991), Internal world models and supervised learning, *in* 'Machine Learning: Proceedings of Eighth Internatinoal Workshop', Morgan Kaufmann, pp. 70–85.

Kakei, S., Hoffman, D. & Strick, P. (1999), 'Muscle and movement representations in the primary motor cortex', *Science* **285**, 2136–2139.

Kakei, S., Hoffman, D. & Strick, P. (2001), 'Direction of action is represented in the ventral premotor cortex', *Nature Neuroscience* **4**, 1020–1025.

Kalman, R. E. (1960), 'A new approach to linear filtering and prediction problems', *Transactions of the ASME - Journal of Basic Engineering* **183**, 35–45.

Kalman, R. E. & Bucy, R. S. (1961), 'New results in linear filtering and prediction theory', *Journal of Basic Engineering, Transactions ASME, Series D* **83**, 95–108.

Kamitani, Y. & Tong, F. (2004), 'Decoding the visual and subjective contents of the human brain', *Nature Neuroscience* **8**, 679.

Kim, S.-J., Koh, K., Lustig, M., Boyd, S. & Gorinevsky, D. (2007), 'A method for large-scale l1-reguarlized least squares', *IEEE Journal on Selected Topics in Signal Processing* **1**(4), 606–617.

Kitagawa, G. (1987), 'Non-Gaussian state-space modeling of non-stationary time series', *Journal of the American Statistical Association* **82**, 1032–1063.

Kitagawa, G. (1996), 'Monte Carlo filter and smoother for non-Gaussian nonlinear state-space models', *Journal of the American Statistical Association* **93**, 1203–1215.

Kitagawa, G. & Gersch, W. (1996), Smoothness priors analysis of time series, *in* 'Lecture Notes in Statistics', Springer-Verlag.

Konolige, K. (2001), Robot motion: probabilistic model; sampling and Gaussian implementations; Markov localization, Technical report, SRI International.

Kramer, S. C. & Sorenson, H. W. (1988), 'Recursive Bayesian estimation using piece-wise constant approximations', *Automatica* **24**(6), 789–801.

Kulback, S. & Leibler, R. A. (1951), 'On information and sufficiency', *The Annals of Mathematical Statistics* **22**(1), 79–86.

Lawrence, N., Seeger, M. & Herbrich, R. (2003), Fast sparse Gaussian process methods: the informative vector machine, *in* 'Proceedings of Advances in Neural Information Processing Systems 15', Vol. 15, MIT Press.

Lee, S., Lee, H., Abeel, P. & Ng, A. (2006), Efficient l1-regularized logistic regresssion, *in* 'Proceedings of the 21st National Conference on Artificial Intelligence', AAAI Press.

Leisink, M. A. R. & Kappen, H. J. (2001), 'A tighter bound for graphical models', *Neural Computation* **13**(9), 2149–2170.

Ljung, L. & Soderstrom, T. (1983), *Theory and Practice of Recursive System Identification*, MIT Press.

Lokhorst, J. (1999), The LASSO and generalized linear models, Technical report, Department of Statistics, University of South Adelaide, South Australia, Australia.

Longford, N. T. (1993), *Random Coefficient Models*, Oxford University Press.

Mackay, D. J. C. (1992), 'A practical Bayesian framework for backpropagation networks', *Neural Computation* **4**, 448–472.

MacKay, D. J. C. (2003), *Information theory, inference, and learning algorithms*, Cambridge University Press.

MacQueen, J. B. (1967), 'Some methods for classification and analysis of multivariate observations', *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* **1**, 281–297.

Masreliez, C. (1975), 'Approximate non-Gaussian filtering with linear state and observation relations', *IEEE Transactions on Automatic Control* **20**, 107–110.

Masreliez, C. & Martin, R. (1977), 'Robust Bayesian estimation for the linear model and robustifying the Kalman filter', *IEEE Transactions on Automatic Control* **22**, 361–371.

Massey, W. (1965), 'Principal component regression in exploratory statistical research', *Journal of the American Statistical Association* **60**, 234–246.

Maybeck, P. S. (1979), *Stochastic models, estimation, and control*, Vol. 141 of *Mathematics in Science and Engineering*, Academic Press.

Meeds, E. & Osindero, S. (2005), An alternative infinite mixture of Gaussian process experts, *in* 'Advances in Neural Information Processing Systems 17', MIT Press.

Meinhold, R. J. & Singpurwalla, N. D. (1989), 'Robustification of Kalman filter models', *Journal of the American Statistical Association* pp. 479–486.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1953), 'Equation of state calculations by fast computing machines', *Journal of Chemical Physics* **21**, 1087–1092.

Minka, T. P. (2001), A family of algorithms for approximate Bayesian inference, PhD thesis, MIT Media Lab, Massachussetts Institute of Technology.

Moore, D. S. & McCabe, G. P. (1999), *Introduction to the Practice of Statistics*, W.H. Freeman & Company.

Morris, J. M. (1976), 'The Kalman filter: A robust estimator for some classes of linear quadratic problems', *IEEE Transactions on Information Theory* **22**, 526–534.

Musallam, S., Corneil, B., Greger, B., Scherberger, H. & Andersen, R. (2004), 'Cognitive control signals for neural prosthetics', *Science* **305**, 258–262.

Myers, K. A. & Tapley, B. D. (1976), 'Adaptive sequential estimation with unknown noise statistics', *IEEE Transactions on Automatic Control* **21**, 520–523.

Neal, R. (1994), Bayesian learning for neural networks, PhD thesis, Department of Computer Science, University of Toronto.

Neal, R. M. (1993), Probabilistic inference using Markov chain Monte Carlo methods, Technical report, University of Toronto.

Neal, R. M. (2003), 'Slice sampling', **31**, 705–767.

Neal, R. M. & Hinton, G. E. (1999), A view of the EM algorithm that justifies incremental, sparse, and other variants, *in* M. I. Jordan, ed., 'Learning in Graphical Models', MIT Press, pp. 355–368.

Ng, A., Jordan, M. & Weiss, Y. (2001), On spectral clustering: analysis and an algorithm, *in* 'Proceedings of Advances in Neural Information Processing Systems 14'.

Nicolelis, M. (2001), 'Actions from thoughts', *Nature* **409**, 403–407.

Ormoneit, D. & Hastie, T. (1999), Optimal kernel shapes for local linear regression, *in* 'Proceedings of Advances in Neural Information Processing Systems', MIT Press.

Paciorek, C. J. & Schervish, M. J. (2004), Non-stationary covariance functions for Gaussian process regression, *in* 'Advances in Neural Information Processing Systems 16', MIT Press.

Peters, J. & Schaal, S. (2008), 'Learning to control in operational space', *International Journal of Robotics Research* **27**, 197–212.

Poggio, T. & Girosi, F. (1990), 'Regularization algorithms for learning that are equivalent to multilayer networks', *Science* **247**, 213–225.

Rao, Y. N., Erdogmus, D., Rao, G. Y. & Principe, J. (2003), Fast error whitening algorithms for system identification and control, *in* 'Proceedings of International Workshop on Neural Networks for Signal Processing', IEEE press, pp. 309–318.

Rao, Y. N. & Principe, J. (2002), Efficient total least squares method for system modeling using minor component analysis, *in* 'Proceedings of International Workshop on Neural Networks for Signal Processing', IEEE press, pp. 259–268.

Rasmussen, C. E. (1996), Evaluation of Gaussian processes and other methods for non-linear regression, PhD thesis, Department of Computer Science, University of Toronto.

Rasmussen, C. E. & Ghahramani, Z. (2002), Infinite mixtures of Gaussian processes, *in* 'Advances in Neural Information Processing Systems 14', MIT Press.

Rissanen, J. (1978), 'Modeling by shortest data description', *Automatica* **14**, 465–471.

Robert, C. P. & Casella, G. (2005), *Monte Carlo statistical methods (Springer texts in Statistics*, Springer.

Rockafellar, R. T. (1972), 'State constraints in convex control problems of bolza', *SIAM Journal on Control and Optimization* **10**, 691–715.

Rustagi, J. S. (1976), 'Variational methods in statistics', *Mathematics in Science and Engineering* **121**.

Ryan, T. P. (1997), *Modern Regression Methods*, Wiley.

Sanguinetti, G., Milo, M., Rattray, M. & Lawrence, N. D. (2005), 'Accounting for probe-level noise in principal component analysis of microarray data', *Bioinformatics* **21**(19), 3748–3754.

Sato, M. (2001), 'Online model selection based on the variational Bayes', *Neural Computation* **13**(7), 1649–1681.

Sato, M. & Ishii, S. (2000), 'Online EM algorithm for the normalized Gaussian network', *Neural Computation* **12**(2), 407–432.

Schaal, S. & Atkeson, C. (1994), Memory-based robot learning, *in* 'Proceedings of the IEEE International Conference on Robotics and Automation', IEEE press, pp. 2928–2933.

Schaal, S. & Atkeson, C. G. (1998), 'Constructive incremental learning from only local information', *Neural Computation* **10**(8), 2047–2084.

Schaal, S., Vijayakumar, S. & Atkeson, C. (1998), Local dimensionality reduction, *in* M. Jordan, M. Kearns & S. Solla, eds, 'Proceedings of Advances in Neural Information Processing Systems 11', MIT Press.

Schick, I. C. & Mitter, S. K. (1994), 'Robust recursive estimation in the presence of heavy-tailed observation noise', *Annals of Statistics* **22**(2), 1045–1080.

Schmidt, A. M. & O'Hagan, A. (2003), 'Bayesian inference for non-stationary spatial covariance structure via spatial deformations', *Journal of Royal Statistical Society. Series B* **65**, 745–758.

Schucany, W. (1995), 'Adaptive bandwidth choice for kernel regression', *Journal of the American Statistical Association* **90**, 535–540.

Sciavicco, L. & Siciliano, B. (1996), *Modeling and control of robot manipulators*, MacGraw-Hill.

Scott, D. W. (2005), Outlier detection and clustering by partial mixture modeling, *in* 'Proceedings in Computational Statistics: 16th Symposium', Physica-Verlag, Heidelberg, pp. 453–465.

Sergio, L. & Kalaska, J. (1998), 'Changes in the temporal pattern of primary motor cortex activity in a directional isometric force versus limb movement task', *Journal of Neurophysiology* **80**, 1577–1583.

Shevade, S. & Keerthi, S. (2003), 'A simple and efficient algorithm for gene selection using sparse logistic regression', *Bioinformatics* **19**(17), 2246–2253.

Shi, J. Q., Murray-Smith, R. & Titterington, D. M. (2005), 'Hierarchical Gaussian process mixtures for regression', *Statistics and Computing* **15**(1), 31–41.

Silverman, B. W. (1985), 'Some aspects of the spline smoothing approach to non-parametric regression curve fitting', *Journal of Royal Statistical Society. Series B* **47**, 1–52.

Smith, A. F. M. & West, M. (1983), 'Monitoring renal transplants: an application of the multiprocess Kalman filter', *Biometrics* **39**, 867–878.

Smola, A. & Scholkoph, B. (2000), Sparse greedy matrix approximation for machine learning, *in* 'Proceedings of the 17th International Conference on Machine Learning', Morgan Kaufmann.

Snelson, E. & Ghahramani, Z. (2006*a*), Sparse Gaussian processes using pseudo-inputs, *in* 'Proceedings of Advances in Neural Information Processing Systems 18', Vol. 18, MIT Press.

Snelson, E. & Ghahramani, Z. (2006*b*), Sparse Gaussian processes using pseudo-inputs, *in* 'Advances in Neural Information Processing Systems 18', MIT Press.

Snelson, E. & Ghahramani, Z. (2006*c*), Variable noise and dimensionality reduction for sparse Gaussian processes, *in* 'Proceedings of Uncertainty in Artificial Intelligence 22', Vol. 22, AUAI Press.

Sorensen, H. W. & Alspach, D. L. (1971), 'Recursive Bayesian estimation using Gaussian sums', *Automatica* **7**, 467–479.

Stone, M. (1974), 'Cross-validatory choice and assessment of statistical predictions', *Journal of Royal Statistical Society. Series B* **36**(1), 111–147.

Stone, M. & Brooks, R. J. (1990), 'Continuum regression: cross-validation sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression', *Journal of Royal Statistical Society. Series B* **52**(2), 237–269.

Strassen, V. (1969), 'Gaussian elimination is not optimal', *Num Mathematik* **13**, 354–356.

Taskar, B., Chatalbashev, V., Koller, D. & Guestrin, C. (2005), Learning structured prediction models: a large margin approach, *in* 'International Conference on Machine Learning'', pp. 896–903.

Taylor, D., Tillery, S. & Schwartz, A. (2002), 'Direct cortical control of 3D neuroprosthetic devices', *Science* **296**, 1829–1932.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the LASSO', *Journal of Royal Statistical Society, Series B* **58**(1), 267–288.

Ting, J., D'Souza, A. & Schaal, S. (2006), Bayesian regression with input noise for high dimensional data, *in* 'Proceedings of the 23rd International Conference on Machine Learning', ACM Press, pp. 937–944.

Ting, J., D'Souza, A. & Schaal, S. (2007), Automatic outlier detection: a Bayesian approach, *in* 'IEEE International Conference on Robotics and Automation'.

Ting, J., D'Souza, A., Yamamoto, K., Yoshioka, T., Hoffman, D., Kakei, S., Sergio, L., Kalaska, J., Kawato, M., Strick, P. & Schaal, S. (2005), Predicting EMG data from M1 neurons with variational Bayesian least squares, *in* 'Proceedings of Advances in Neural Information Processing Systems 18', MIT Press.

Ting, J., D'Souza, A., Yamamoto, K., Yoshioka, T., Hoffman, D., Kakei, S., Sergio, L., Kalaska, J., Kawato, M., Strick, P. & Schaal, S. (2008), 'Variational Bayesian least squares: an application to brain-machine interface data', *Neural Networks: Special Issue on Neuroinformatics* **21**(8), 1112–1131.

Ting, J., Kalakrishnan, M., Vijayakumar, S. & Schaal, S. (2008), Bayesian kernel shaping for learning control, *in* 'Proceedings of Advances in Neural Information Processing Systems 21', MIT Press.

Ting, J., Mistry, M., Peters, J., Schaal, S. & Nakanishi, J. (2006), A Bayesian approach to nonlinear parameter identification for rigid body dynamics, *in* 'Proceedings of Robotics: Science and Systems'.

Ting, J., Theodorou, E. & Schaal, S. (2007), Learning an outlier-robust Kalman filter, *in* 'European Conference on Machine Learning', Vol. 4701 of *Lecture Notes in Computer Science*, Springer, pp. 748–756.

Tipping, M. E. (2001), 'Sparse Bayesian learning and the Relevance Vector Machine', *Journal of Machine Learning Research* **1**, 211–244.

Todorov, E. (2000), 'Direct cortical control of muscle activation in voluntary arm movements: a model', *Nature Neuroscience* **3**, 391–398.

Tresp, V. (2000), Mixtures of Gaussian processes, *in* 'Advances in Neural Information Processing Systems 13', MIT Press.

Tukey, J. W. (1960), A survey of sampling from contaminated distributions, *in* I. Olkin, ed., 'Contributions to Probability and Statistics', Stanford University Press, pp. 448–485.

Van Huffel, S. & Lemmerling, P. (2002), *Total Least Squares and Errors-in-Variables Modeling: Analysis, Algorithms and Applications*, Kluwer Academic Publishers, Dordrecht, Netherlands.

Van Huffel, S. & Vanderwalle, J. (1991), 'The total least squares problem: computational aspects and analysis', *Society for Industrial and Applied Mathematics* .

Vijayakumar, S., D'Souza, A. & Schaal, S. (2005), 'Incremental online learning in high dimensions', *Neural Computation* **17**, 1–33.

Wainwright, W. J., Jaakkola, T. & Willsky, A. S. (2002), A new class of upper bounds on the log partition function, *in* 'Proceedings of Uncertainty in Artificial Intelligence 18', Vol. 18, Morgan Kaufmann.

Wessberg, J. & Nicolelis, M. (2004), 'Optimizing a linear algorithm for real-time robotic control using chronic cortical ensemble recordings in monkeys', *Journal of Cognitive Neuroscience* **16**, 1022–1035.

West, M. (1981), 'Robust sequential approximate Bayesian estimation', *Journal of the Royal Statistical Society, Series B* **43**, 157–166.

West, M. (1982), Aspects of recursive Bayesian estimation, PhD thesis, Department of Mathematics, University of Nottingham.

Williams, C. K. I. & Rasmussen, C. E. (1995), Gaussian processes for regression, *in* D. S. Touretzky, M. C. Mozer & M. E. Hasselmo, eds, 'Proceedings of Advances in Neural Information Processing Systems 8', Vol. 8, MIT Press.

Wold, H. (1975), Soft modeling by latent variables: the nonlinear iterative partial least squares approach, *in* J. Gani, ed., 'Perspectives in probability and statistics, papers in honor of M. S. Bartlett', Academic Press.

Wolpaw, J. & McFarland, D. (2004), 'Control of a two-dimensional movement signal by a noninvasive brain-computer interface in humans', *Proceedings of the National Academy of Sciences* **101**, 17849–17854.

Y. Shen, A. N. & Seeger, M. (2006), Fast Gaussian process regression using KD-trees, *in* 'Proceedings of Advances in Neural Information Processing Systems 18', Vol. 18, MIT Press.

Yedidia, J. S., Freeman, W. T. & Weiss, Y. (2001), Bethe free energy, Kikuchi approximations and belief propagation algorithms, Technical Report TR2001-16, Mitsubishi Electric Research Laboratories.

Appendix A

Variational Bayesian Least Squares

A.1 Derivation of Variational Bayesian Least Squares

Consider the complete log likelihood of the model in Section 2.2.2, i.e., Eq. (2.15). We know that maximizing the lower bound implies that the functional $F(Q, \phi)$ over the space of probability distributions $Q(\mathbf{v}_H)$, where $\mathbf{v}_H$ are unobserved random variables. The calculus of variations, e.g., (Rustagi 1976), allows for the derivation of general updates for distributions that are individually factored. For the case where we assume individually factored distributions of the random variables:

$$Q(\mathbf{v}_H) = Q_1(\mathbf{v}_1)Q_2(\mathbf{v}_2)\cdots Q_n(\mathbf{v}_n),$$

the solution for the individual posterior distributions that maximizes the function $F(Q, \phi)$ under the factorization assumption is:

$$Q_j(\mathbf{v}_j) = \frac{\exp\left\langle \log p(\mathbf{x}_D, \mathbf{v}_H; \phi)\right\rangle_{Q_{k\neq j}}}{\int \exp\left\langle \log p(\mathbf{x}_D, \mathbf{v}_H; \phi)\right\rangle_{Q_{k\neq j}} d\mathbf{v}_j}$$

or equivalently:

$$\log Q_j(\mathbf{v}_j) = \left\langle \log p(\mathbf{x}_D, \mathbf{v}_H; \phi)\right\rangle_{Q_{k\neq j}} + \text{const}_{\mathbf{v}_j} \tag{A.1}$$

where $\mathbf{x}_D$ is the observed data, $\langle\cdot\rangle_{Q_{k\neq j}}$ denotes the expectation taken with respect to all distributions Q_k except for Q_j, and $j = 1, 2, ..., n$.

For the scenario where we do not assume a full factorized assumption of $\mathbf{v}_H$ and instead, assume, for example:

$$Q(\mathbf{v}_H) = Q_{12}(\mathbf{v}_1, \mathbf{v}_2)Q_3(\mathbf{v}_3)\cdots Q_n(\mathbf{v}_n)$$

we would need to infer the joint distribution of $Q_{12}(\mathbf{v}_1, \mathbf{v}_2)$. One approach of solving this is to assume that the joint distribution factorizes into $Q_1(\mathbf{v}_1|\mathbf{v}_2)Q_2(\mathbf{v}_2)Q_3(\mathbf{v}_3)\cdots Q_n(\mathbf{v}_n)$. The resulting posterior distribution for $Q_1(\mathbf{v}_1|\mathbf{v}_2)$ can be derived using Eq. (A.1) above. However, when maximizing the functional with respect to Q_2, we would need to include

terms that have a dependency on $\mathbf{v}_2$ in the numerator (or, equivalently, we could use Eq. (A.1) but then marginalize out $\mathbf{v}_1$):

$$Q_2(\mathbf{v}_2) \propto \exp\left(\langle \log p(\mathbf{x}_D, \mathbf{v}_H; \phi)\rangle_{Q_{k\neq 2}} - \int Q_1(\mathbf{v}_1|\mathbf{v}_2) \log Q_1(\mathbf{v}_1|\mathbf{v}_2) d\mathbf{v}_1\right)$$

or equivalently:

$$\log Q_2(\mathbf{v}_2) = \langle \log p(\mathbf{x}_D, \mathbf{v}_H; \phi)\rangle_{Q_{k\neq 2}} + \text{Entropy}\left\{Q_1(\mathbf{v}_1|\mathbf{v}_2)\right\} + \text{const} \tag{A.2}$$

With the above in mind, we can derive the final EM updates for the model in Section 2.2.2, assuming that:

$$\begin{aligned} Q(\mathbf{b}, \boldsymbol{\alpha}, \mathbf{Z}) &= Q(\mathbf{b}, \boldsymbol{\alpha}) Q(\mathbf{Z}) \\ &= Q(\mathbf{b}|\boldsymbol{\alpha}) Q(\boldsymbol{\alpha}) Q(\mathbf{Z}) \end{aligned}$$

We can then infer the posterior distributions analytically:

- For $Q(\mathbf{b}, \boldsymbol{\alpha})$:

$$\begin{aligned} \log Q(\mathbf{b}, \boldsymbol{\alpha}) = &\frac{N}{2} \sum_{m=1}^{d} \log \alpha_m - \sum_{m=1}^{d} \frac{\alpha}{2\psi_{zm}} \sum_{i=1}^{N} \left\langle (z_{im} - b_m x_{im})^2 \right\rangle \\ &+ \frac{1}{2} \sum_{m=1}^{d} \log \alpha_m - \frac{1}{2} \sum_{m=1}^{d} \alpha_m b_m^2 \\ &+ \sum_{m=1}^{d} (a_{\alpha_m,0} - 1) \log \alpha_m - \sum_{m=1}^{d} b_{\alpha_m,0} \alpha_m + \text{const}_{\boldsymbol{\alpha},\mathbf{b}} \end{aligned}$$

  From the above equation, we can deduce:

$$\begin{aligned} Q(\mathbf{b}, \boldsymbol{\alpha}) &= Q(\boldsymbol{\alpha}) \prod_{m=1}^{d} Q(b_m|\alpha_m) \\ Q(b_m|\alpha_m) &= \text{Normal}\left(\langle b_m|\alpha_m\rangle, \sigma^2_{b_m|\alpha_m}\right) \\ \sigma^2_{b_m|\alpha_m} &= \frac{\psi_{zm}}{\langle \alpha_m \rangle} \left(\sum_{i=1}^{N} x_{im}^2 + \psi_{zm}\right)^{-1} \\ \langle b_m|\alpha_m\rangle &= \left(\sum_{i=1}^{N} x_{im}^2 + \psi_{zm}\right)^{-1} \left(\sum_{i=1}^{N} \langle z_{im}\rangle x_{im}\right) \end{aligned} \tag{A.3}$$

  Note that the posterior mean of $b_m|\alpha_m$ in Eq. (A.3) is independent of α_m. As a result, we can re-write the conditional posterior of b_m as:

$$\log Q(\mathbf{b}, \boldsymbol{\alpha}) = -\frac{1}{2} \sum_{m=1}^{d} \left[\frac{1}{\sigma^2_{b_m|\alpha_m}} \left(b_m - \langle b_m|\alpha_m\rangle\right)^2 - \frac{\langle b_m|\alpha_m\rangle^2}{\sigma^2_{b_m|\alpha_m}} + \frac{\alpha_m}{\psi_{zm}} \sum_{i=1}^{N} \langle z_{im}^2\rangle \right]$$

$$+\sum_{m=1}^{d}\left[\left(a_{\alpha_m,0}+\frac{N}{2}+\frac{1}{2}-1\right)\log\alpha_m-b_{\alpha_m,0}\alpha_m\right]+\text{const}_{\boldsymbol{\alpha},\mathbf{b}}$$

Taking the exponent of the above expression and integrating out $\mathbf{b}$, we can derive the following posterior distribution for $\boldsymbol{\alpha}$:

$$\begin{aligned}\log Q(\boldsymbol{\alpha}) &= -\frac{1}{2}\sum_{m=1}^{d}\left[-\log 2\pi\sigma^2_{b_m|\alpha_m}+\frac{\alpha_m}{\psi_{zm}}\sum_{i=1}^{N}\left\langle z_{im}^2\right\rangle-\frac{\langle b_m|\alpha_m\rangle^2}{\sigma^2_{b_m|\alpha_m}}\right]\\ &\quad+\left(a_{\alpha_m,0}+\frac{N+1}{2}-1\right)\log\alpha_m-b_{\alpha_m,0}+\text{const}_{\boldsymbol{\alpha}}\\ &= -\frac{1}{2}\sum_{m=1}^{d}\left[\frac{\alpha_m}{\psi_{zm}}\sum_{i=1}^{N}\left\langle z_{im}^2\right\rangle-\frac{\langle b_m|\alpha_m\rangle^2}{\sigma^2_{b_m|\alpha_m}}\right]\\ &\quad+\left(a_{\alpha_m,0}+\frac{N}{2}-1\right)\log\alpha_m-b_{\alpha_m,0}+\text{const}_{\boldsymbol{\alpha}}\end{aligned}$$

From the above expression, we can infer the posterior distribution over $\boldsymbol{\alpha}$ is:

$$\begin{aligned}Q(\boldsymbol{\alpha}) &= \prod_{m=1}^{d}\text{Gamma}\left(\hat{a}_{\alpha_m},\hat{b}_{\alpha_m}\right)\\ \hat{a}_{\alpha_m} &= a_{\alpha_m,0}+\frac{N}{2}\\ \hat{b}_{\alpha_m} &= b_{\alpha_m,0}+\frac{1}{2\psi_{zm}}\left\{\sum_{i=1}^{N}\left\langle z_{im}^2\right\rangle-\left(\sum_{i=1}^{N}x_{im}^2+\psi_{zm}\right)^{-1}\left(\sum_{i=1}^{N}\left\langle z_{im}\right\rangle x_{im}\right)^2\right\}\end{aligned}$$

- For $Q(\mathbf{Z})$:

$$\log Q(\mathbf{Z}) = -\frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i-\mathbf{1}^T\mathbf{z}_i\right)^2-\sum_{m=1}^{d}\left\langle\frac{\alpha_m}{2\psi_y}\sum_{i=1}^{N}\left(z_{im}-b_m x_{im}\right)^2\right\rangle$$

From the above, we can infer the posterior distribution of $\mathbf{Z}$ to be:

$$\begin{aligned}Q(\mathbf{Z}) &= \prod_{m=1}^{d}\text{Normal}\left(\left\langle\mathbf{z}_i\right\rangle,\boldsymbol{\Sigma}_{\mathbf{z}}\right)\\ \boldsymbol{\Sigma}_{\mathbf{z}} &= \left(\frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T+\boldsymbol{\Psi}_z^{-1}\left\langle\mathbf{A}\right\rangle\right)^{-1}=\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}-\frac{\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}\mathbf{1}^T\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}}{\psi_y+\mathbf{1}^T\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}}\\ \left\langle\mathbf{z}_i\right\rangle &= \boldsymbol{\Sigma}_{\mathbf{z}}\left(\frac{1}{\psi_y}\mathbf{1}y_i+\boldsymbol{\Psi}_z^{-1}\left\langle\mathbf{A}\right\rangle\left\langle\mathbf{B}|\mathbf{A}\right\rangle\mathbf{x}_i\right)\\ &= \left(\frac{\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}}{\psi_y+\mathbf{1}^T\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}}\right)y_i+\left(\left\langle\mathbf{B}|\mathbf{A}\right\rangle-\frac{\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}\mathbf{1}^T\left\langle\mathbf{B}|\mathbf{A}\right\rangle}{\psi_y+\mathbf{1}^T\boldsymbol{\Psi}_z\left\langle\mathbf{A}\right\rangle^{-1}\mathbf{1}}\right)\mathbf{x}_i\end{aligned}$$

$$\langle z_{im} \rangle = \langle b_m | \alpha_m \rangle \, x_{im} + \frac{1}{s} \frac{\psi_{zm}}{\langle \alpha_m \rangle} \left(y_i - \langle \mathbf{b} | \boldsymbol{\alpha} \rangle^T \mathbf{x}_i \right)$$

$$\left\langle z_{im}^2 \right\rangle = \frac{\psi_{zm}}{\langle \alpha_m \rangle} - \frac{1}{s} \left(\frac{\psi_{zm}}{\langle \alpha_m \rangle} \right)^2 + \langle z_{im} \rangle^2$$

$$\mathbf{1}^T \langle \mathbf{z}_i \rangle = \frac{1}{2} \left(\sum_{m=1}^{d} \frac{\psi_{zm}}{\langle \alpha_m \rangle} \right) y_i + \left(1 - \frac{1}{s} \left(\sum_{m=1}^{d} \frac{\psi_{zm}}{\langle \alpha_m \rangle} \right) \right) \langle \mathbf{b} | \boldsymbol{\alpha} \rangle^T \mathbf{x}_i$$

$$\mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} = \sum_{m=1}^{d} \frac{\psi_{zm}}{\langle \alpha_m \rangle} - \frac{1}{s} \left(\sum_{m=1}^{d} \frac{\psi_{zm}}{\langle \alpha_m \rangle} \right)^2 + \left(\mathbf{1}^T \langle \mathbf{z}_i \rangle \right)^2$$

where $\boldsymbol{\Psi}^{-1} \langle \mathbf{A} \rangle = \text{diag}[\langle \alpha_m \rangle / \psi_{zm}]$ and $s = \psi_y + \mathbf{1}^T \boldsymbol{\Psi}_z \langle \mathbf{A} \rangle^{-1} \mathbf{1}$.

A.2 Pseudocode for Variational Bayesian Least Squares

The pseudocode for VBLS is listed below 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:

$$\begin{aligned} \log p(\mathbf{y} | \mathbf{X}; \phi) &\geq \int Q(\boldsymbol{\theta}) \log \frac{p(\mathbf{y}, \boldsymbol{\theta} | \mathbf{X}; \phi)}{Q(\boldsymbol{\theta})} d\boldsymbol{\theta} \\ &\geq \int Q(\boldsymbol{\theta}) \log p(\mathbf{y}, \boldsymbol{\theta} | \mathbf{X}; \phi) d\boldsymbol{\theta} - \int Q(\boldsymbol{\theta}) \log Q(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &\geq \langle \log p(\mathbf{y}, \boldsymbol{\theta} | \mathbf{X}; \phi \rangle_{Q(\boldsymbol{\theta})} - \int Q(\boldsymbol{\theta}) \log Q(\boldsymbol{\theta}) d\boldsymbol{\theta} \end{aligned} \tag{A.4}$$

where Eq. (A.4) simplifies to:

$$
\begin{aligned}
&\log p(\mathbf{y}|\mathbf{X};\phi) \geq \\
&\quad -\frac{N}{2}\log\psi_y - \frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i^2 - 2y_i\mathbf{1}^T\left\langle\mathbf{z}_i\right\rangle + \mathbf{1}^T\left\langle\mathbf{z}_i\mathbf{z}_i^T\right\rangle\mathbf{1}\right) \\
&\quad -\frac{N}{2}\sum_{m=1}^{d}\log\psi_{zm} - \sum_{m=1}^{d}\frac{\langle\alpha_m\rangle}{2\psi_{zm}}\sum_{i=1}^{N}\left(\left\langle z_{im}^2\right\rangle - 2\left\langle z_{im}\right\rangle\left\langle b_m|\alpha_m\right\rangle x_{im} + \left\langle\left(b_m|\alpha_m\right)^2\right\rangle x_{im}^2\right) \\
&\quad -\frac{1}{2}\sum_{m=1}^{d}\langle\alpha_m\rangle\left\langle\left(b_m|\alpha_m\right)^2\right\rangle - \frac{N-1}{2}\sum_{m=1}^{d}\log\hat{b}_{\alpha_m} - \hat{a}_{\alpha_m} \\
&\quad -\frac{1}{2}\log|\mathbf{\Sigma}_z^{-1}| - \sum_{m=1}^{d}\log\hat{b}_{\alpha_m} + \frac{1}{2}\sum_{m=1}^{d}\langle\alpha_m\rangle\left(\sigma_{b_m|\alpha_m}^2 + 1\right) + \text{const}_{\mathbf{y},\phi}
\end{aligned}
\tag{A.5}
$$

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.

0: **Initialization:** $a_{\boldsymbol{\alpha},0} = 10^{-8}\mathbf{1}$, $b_{\boldsymbol{\alpha},0} = 10^{-8}\mathbf{1}$; 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. (2.16) to (2.22)
4: Perform the M-step: Calculate Eqs. (2.23) and (2.24)
5: Monitor the lower bound to the incomplete log likelihood, Eq. (A.5), to see if the likelihood tolerance t has been reached
6: **until** convergence of Eq. (A.5)

Algorithm 1: Pseudocode for variational Bayesian least squares

A.3 EM Update Equations for Real-time Implementation

The incremental EM update equations for the kth time step, when data sample $\{\mathbf{x}_k, y_k\}$ is available, are listed below. Note that ${\psi_y}^{(k-1)}$ denotes the value of ψ_y at the $(k-1)$th time step, and the same notation is used for all other parameters.

E-step:

$$
\begin{aligned}
\mathbf{\Sigma}_z^{(k)} &= \mathbf{\Psi}_z^{(k-1)}\left\langle\mathbf{A}^{-1}\right\rangle^{(k-1)} - \frac{\mathbf{\Psi}_z^{(k-1)}\left\langle\mathbf{A}^{-1}\right\rangle^{(k-1)}\mathbf{1}\mathbf{1}^T\mathbf{\Psi}_z^{(k-1)}\left\langle\mathbf{A}^{-1}\right\rangle^{(k-1)}}{s^{(k)}} \\
\langle\mathbf{z}\rangle^{(k)} &= \left(\frac{\mathbf{\Psi}_z^{(k-1)}\left\langle\mathbf{A}^{-1}\right\rangle^{(k-1)}\mathbf{1}}{s^{(k)}}\right)y_k + \langle\mathbf{B}\rangle^{(k-1)}\mathbf{x}_k
\end{aligned}
\tag{A.6}
$$

$$- \frac{\boldsymbol{\Psi}_z^{(k-1)} \left\langle \mathbf{A}^{-1} \right\rangle^{(k-1)} \mathbf{1}\mathbf{1}^T \langle \mathbf{B} \rangle^{(k-1)}}{s^{(k)}} \mathbf{x}_k \tag{A.7}$$

$$\left(\sigma_{b_m}^2\right)^{(k)} = \frac{\psi_{zm}^{(k-1)}}{\langle \alpha_m \rangle^{(k-1)}} \left(\boldsymbol{\Sigma}_{x^2}^{(k)} + \psi_{zm}^{(k-1)}\right)^{-1} \tag{A.8}$$

$$\langle b_m \rangle^{(k)} = \left(\frac{\boldsymbol{\Sigma}_{x^2}^{(k)}}{\boldsymbol{\Sigma}_{x^2}^{(k)} + \psi_{zm}^{(k-1)}}\right) \langle b_m \rangle^{(k-1)} + \frac{\psi_{zm}^{(k-1)}}{\langle \alpha_m \rangle^{(k-1)}} \frac{\left(\boldsymbol{\Sigma}_{yx}^{(k)} - \boldsymbol{\Sigma}_{xx^T}^{(k)} \langle \mathbf{b} \rangle^{(k-1)}\right)}{\left(\boldsymbol{\Sigma}_{x^2}^{(k)} + \psi_{zm}^{(k-1)}\right) s^{(k)}} \tag{A.9}$$

$$\langle \alpha_m \rangle^{(k)} = \frac{a_{\alpha_m,0} + 0.5\boldsymbol{\Sigma}_N^{(k)}}{b_{\alpha_m,0} + \frac{1}{2\psi_{zm}^{(k-1)}} \left\{ \boldsymbol{\Sigma}_N^{(k)} \boldsymbol{\Sigma}_z{}^{(k)} + \boldsymbol{\Sigma}_{\langle z^2 \rangle}^{(k)} - \left(\boldsymbol{\Sigma}_{x^2}^{(k)} + \psi_{zm}^{(k-1)}\right)^{-1} \left(\boldsymbol{\Sigma}_{\langle z \rangle x}^{(k)}\right)^2 \right\}} \tag{A.10}$$

M-step:

$$\psi_y^{(k)} = \frac{1}{\boldsymbol{\Sigma}_N^{(k)}} \left(1 - \frac{\boldsymbol{\Psi}_z^{(k-1)} \left\langle \mathbf{A}^{-1} \right\rangle^{(k-1)} \mathbf{1}}{s^{(k)}}\right)^2 \left(\boldsymbol{\Sigma}_{y^2}^{(k)} - 2\left(\langle \mathbf{b} \rangle^{(k-1)}\right)^T \boldsymbol{\Sigma}_{yx}^{(k)} + \left(\langle \mathbf{b} \rangle^{(k-1)}\right)^T \boldsymbol{\Sigma}_{xx^T}^{(k)} \langle \mathbf{b} \rangle^{(k-1)}\right) + \mathbf{1}^T \boldsymbol{\Sigma}_z^{(k)} \mathbf{1} \tag{A.11}$$

$$\psi_{zm}^{(k)} = \frac{1}{\langle \alpha_m \rangle^{(k)}} \left(\frac{\psi_{zm}{}^{(k-1)}}{s^{(k)}}\right)^2 \frac{\left(\boldsymbol{\Sigma}_{y^2}^{(k)} - 2\left(\langle \mathbf{b} \rangle^{(k)}\right)^T \boldsymbol{\Sigma}_{yx}^{(k)} + \left(\langle \mathbf{b} \rangle^{(k)}\right)^T \boldsymbol{\Sigma}_{xx^T}^{(k)} \langle \mathbf{b} \rangle^{(k)}\right)}{\boldsymbol{\Sigma}_N^{(k)}} + \langle \alpha_m \rangle^{(k)} \left(\sigma_{b_m}^2\right)^{(k)} \frac{\boldsymbol{\Sigma}_{x^2}^{(k)}}{\boldsymbol{\Sigma}_N^{(k)}} + \langle \alpha_m \rangle^{(k)} \boldsymbol{\Sigma}_z^{(k)} \tag{A.12}$$

where $\left\langle \mathbf{A}^{-1} \right\rangle = \langle \mathbf{A} \rangle^{-1}$ (since $\mathbf{A}$ is a diagonal matrix consisting of the diagonal vector $\boldsymbol{\alpha}$) and:

$$s^{(k)} = \psi_y^{(k-1)} + \mathbf{1}^T \boldsymbol{\Psi}_z^{(k-1)} \left\langle \mathbf{A}^{-1} \right\rangle^{(k-1)} \mathbf{1} \tag{A.13}$$

Additionally, the sufficient statistics used in Eqs. (A.6) to (A.12) are:

$$\begin{aligned}
\boldsymbol{\Sigma}_{x^2}^{(k)} &= \boldsymbol{\Sigma}_{x^2}^{(k-1)} + x_{im}^2 & \boldsymbol{\Sigma}_{y^2}^{(k)} &= \boldsymbol{\Sigma}_{y^2}^{(k-1)} + y_i^2 \\
\boldsymbol{\Sigma}_{yx}^{(k)} &= \boldsymbol{\Sigma}_{yx}^{(k-1)} + y_i x_{im} & \boldsymbol{\Sigma}_{xx^T}^{(k)} &= \boldsymbol{\Sigma}_{xx^T}^{(k-1)} + \mathbf{x}_i \mathbf{x}_i^T \\
\boldsymbol{\Sigma}_N^{(k)} &= \lambda \boldsymbol{\Sigma}_N^{(k-1)} + 1 & \boldsymbol{\Sigma}_{\langle z \rangle x}^{(k)} &= \lambda \boldsymbol{\Sigma}_{\langle z \rangle x}^{(k-1)} + \langle z_{im} \rangle x_{im} \\
\boldsymbol{\Sigma}_{\langle z^2 \rangle}^{(k)} &= \lambda \boldsymbol{\Sigma}_{\langle z^2 \rangle}^{(k-1)} + \langle z_{im} \rangle^2
\end{aligned}$$

with certain sufficient statistics discounted by λ, as necessary.

Appendix B

EMG Prediction from Neural Data: Plots

B.1 Construction of Cross-validation Sets

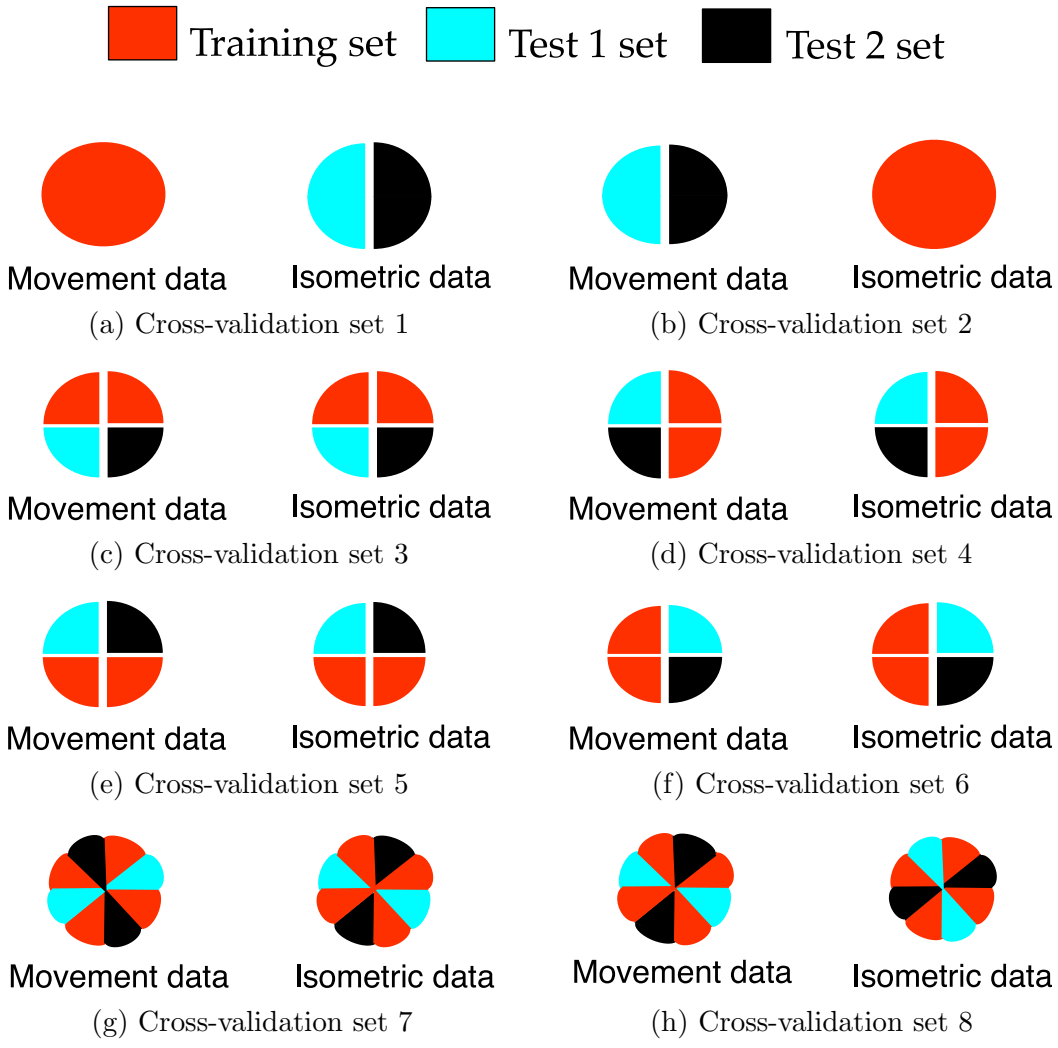
The next four figures, Figures B.1 to B.3, show how cross-validation sets are constructed for each of the data sets discussed in Section 2.4.2.

In Figure B.1, for each type of force applied by the monkey to the manipulandum, there are eight possible directions that the manipulandum could have been moved. The data is from (Sergio & Kalaska 1998) and involves neural firing from the M1 cortex. Each circle shown in the figure is partitioned into eight equal portions, corresponding to the eight directional movements and numbered in increasing order (clockwise) starting from one.

In Figure B.2, there are also eight possible directional movements for each of the three wrist positions[1]. The data is from Kakei et al. (1999) and involves neural firing data from the M1 cortex. As with the previous figure, each circle is partitioned into eight equal sections, corresponding to the eight directional movements and numbered in increasing order (clockwise) starting from one.

Figure B.3 shows the construction of cross-validation sets, which are done in a similar fashion to Figure B.2—except it involves neural firing data from the PM cortex, taken from the Kakei et al. (2001) data set.

[1]The three possible wrist positions are i) supinated, ii) pronated and iii) midway in between the two positions.



(a) Cross-validation set 1

(b) Cross-validation set 2

(c) Cross-validation set 3

(d) Cross-validation set 4

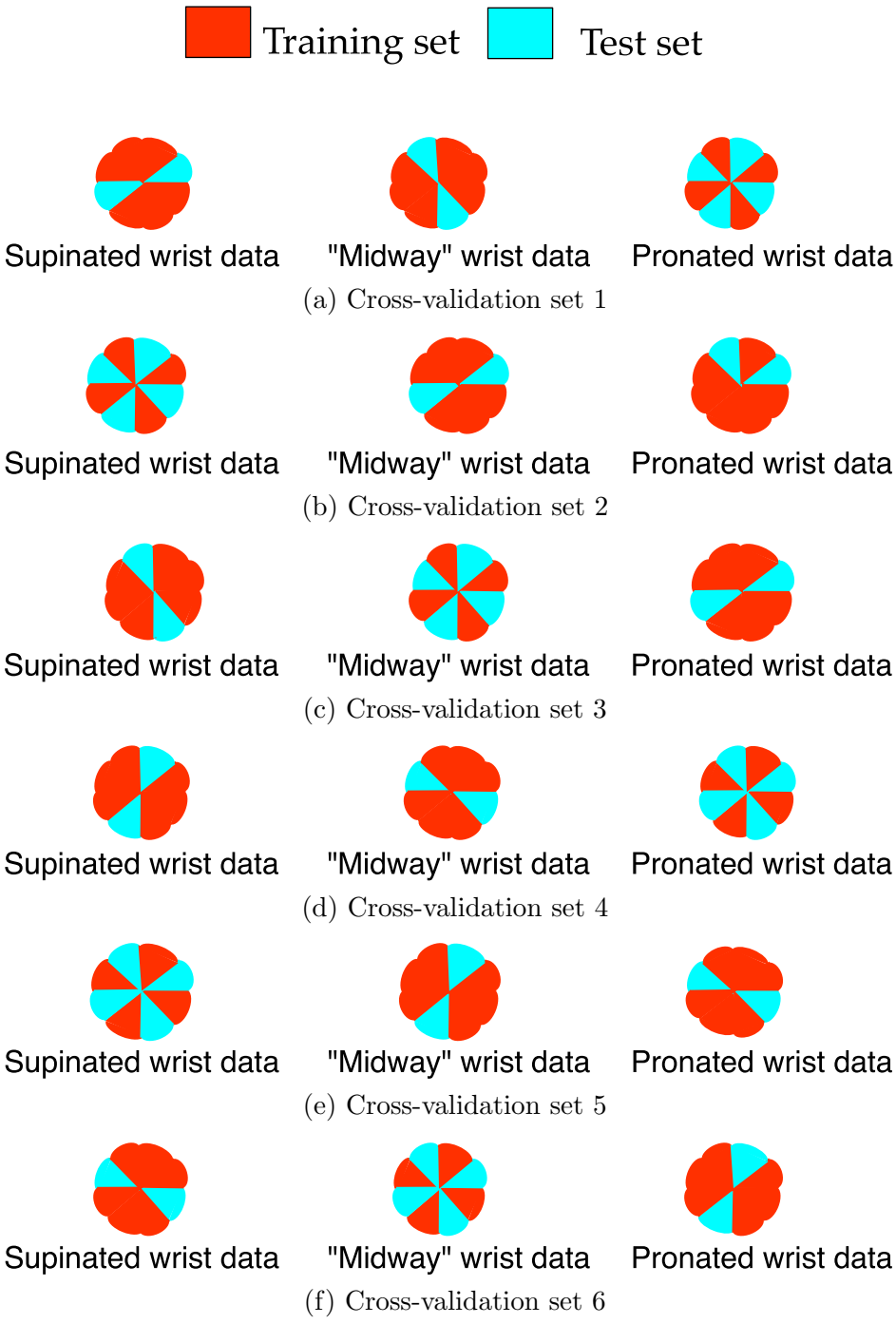
(e) Cross-validation set 5

(f) Cross-validation set 6

(g) Cross-validation set 7

(h) Cross-validation set 8

Figure B.1: Cross-validation splits for the Sergio & Kalaska (1998) M1 neural data set.



(a) Cross-validation set 1

(b) Cross-validation set 2

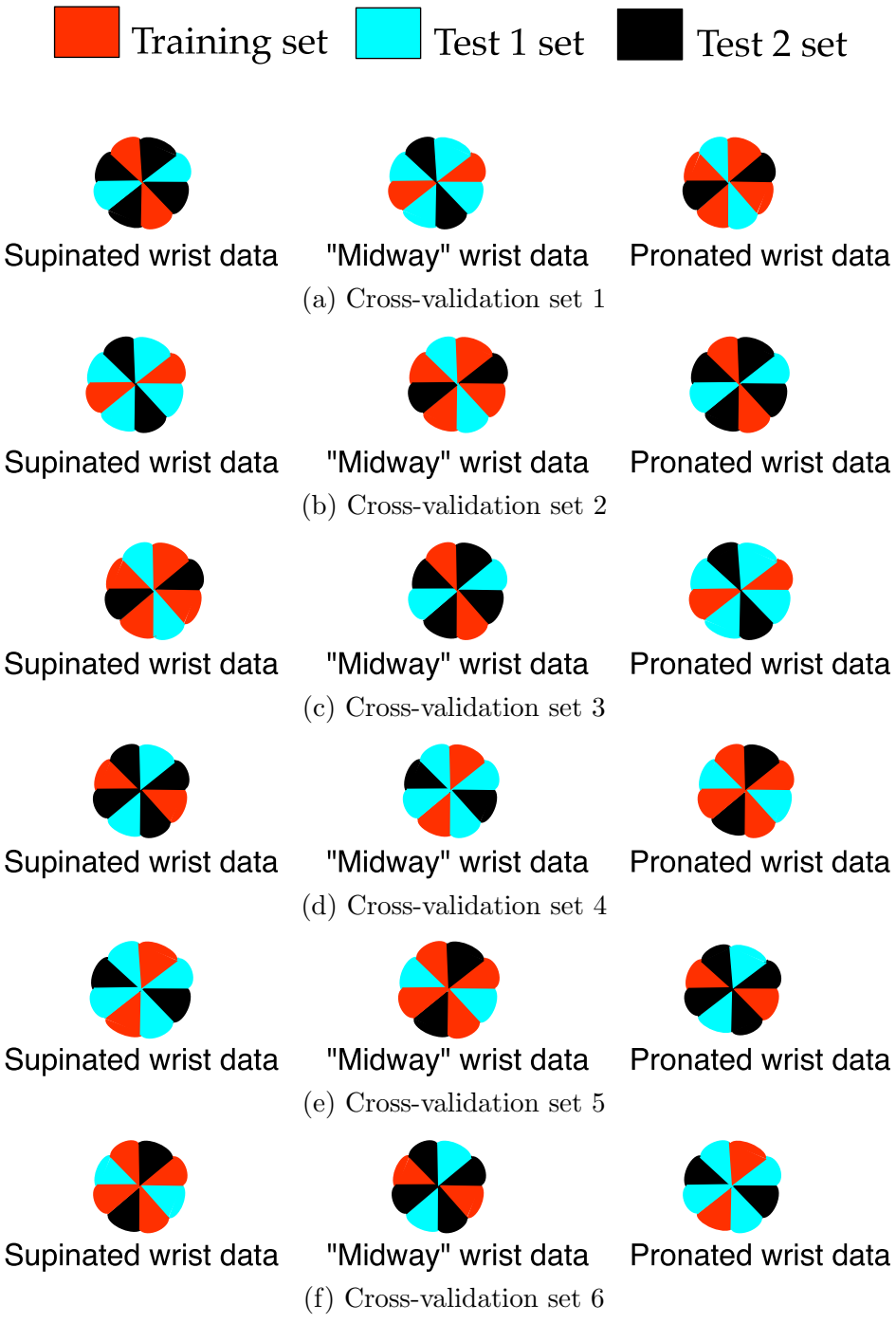
(c) Cross-validation set 3

(d) Cross-validation set 4

(e) Cross-validation set 5

(f) Cross-validation set 6

Figure B.2: Cross-validation splits for the Kakei et al. (1999) M1 neural data set.



(a) Cross-validation set 1

(b) Cross-validation set 2

(c) Cross-validation set 3

(d) Cross-validation set 4

(e) Cross-validation set 5

(f) Cross-validation set 6

Figure B.3: Cross-validation splits for the Kakei et al. (2001) PM neural data set.

B.2 Plots of Percentage Matches with Baseline Study

The following three plots, Figures B.4 to B.6, show the percentage matches of neurons found to be relevant by each algorithm, as compared to those found by the baseline study (Modelsearch). Percentage matches are shown for each muscle for all three neural data sets.

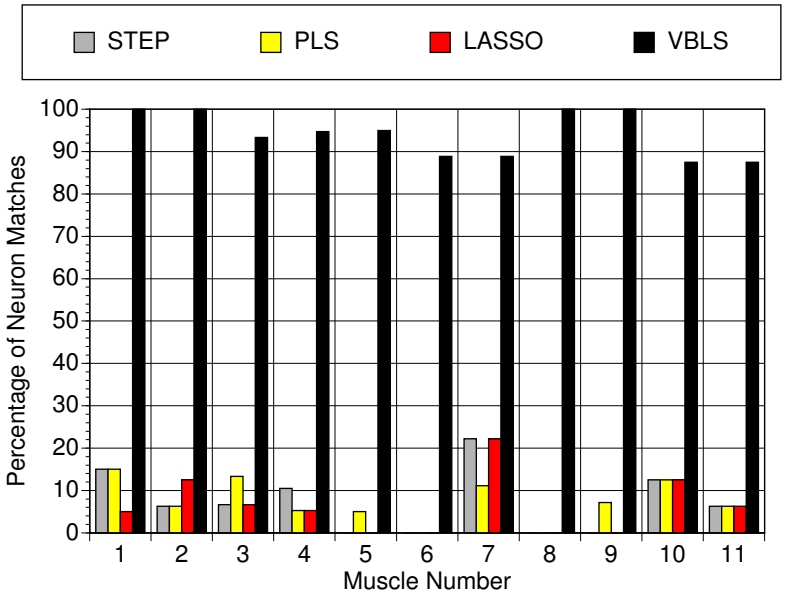


Figure B.4: Percentage of M1 neuron matches found by each algorithm, as compared to those found by the baseline study (ModelSearch), shown for each muscle in the Sergio & Kalaska (1998) data set.

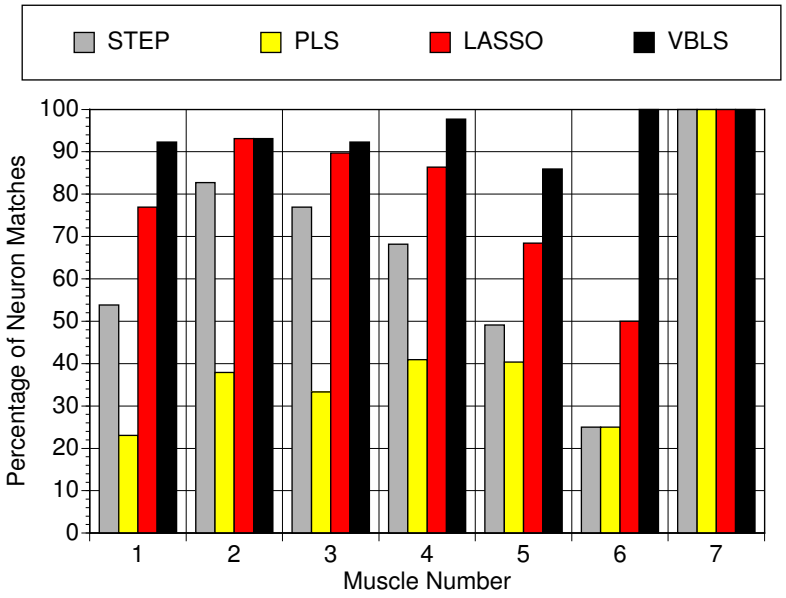


Figure B.5: Percentage of M1 neuron matches found by each algorithm, as compared to those found by the baseline study (ModelSearch), shown for each muscle in the Kakei et al. (1999) data set.

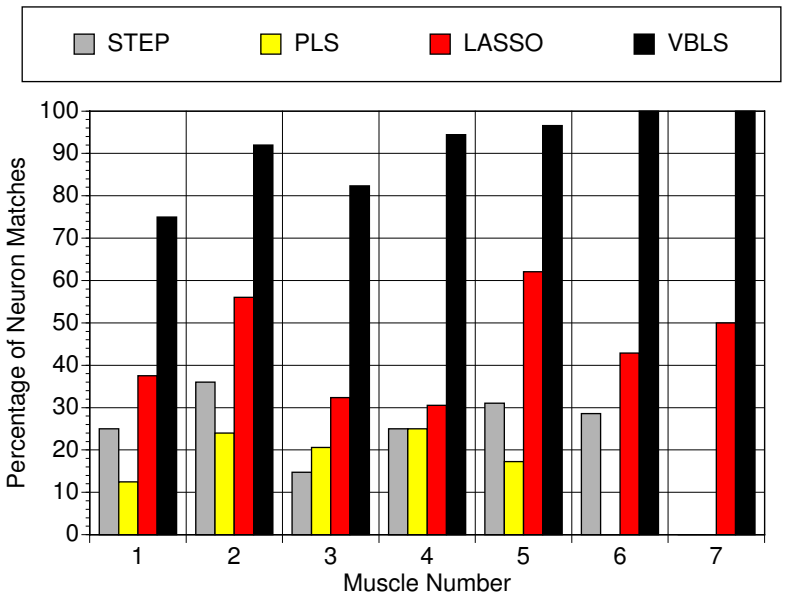
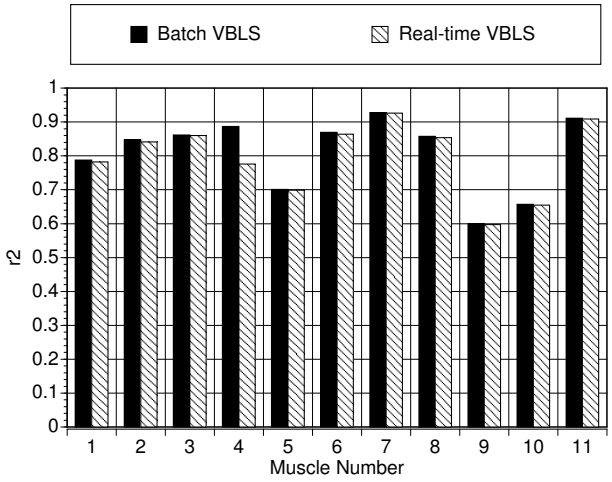
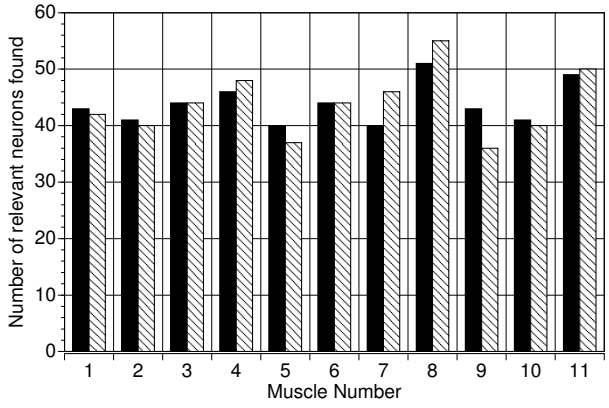


Figure B.6: Percentage of PM neuron matches found by each algorithm, as compared to those found by the baseline study (ModelSearch), shown for each muscle in the Kakei et al. (2001) data set.

B.3 Real-time Analysis for Brain-Machine Interfaces



(a) r^2 values.



(b) Number of relevant M1 neurons found.

Figure B.7: Coefficient of determination values, $r^2 = 1 - \text{nMSE}$, and number of relevant neurons found by VBLS—both the batch and real-time versions—for the Sergio & Kalaska (1998) data set. For the real-time, incremental version of VBLS, the number of relevant neurons found in the last time step is shown.

B.4 Plots of EMG Traces and Velocities

The following six plots show the EMG traces and (x, y) velocities for various muscles under certain wrist conditions in the Kakei et al. (1999) and Kakei et al. (2001) data sets.

The first two plots, Figures B.8(a) and B.9(a), show the observed and predicted EMG traces for the ECRB muscle while in the supinated wrist condition, given firing from M1 and PM neurons, respectively. For each figure, the center plot shows the trajectories in eight different directions—in the (x, y) plane—taken by the hand. Each of the eight plots surrounding the center plot shows the EMG traces over time for each hand trajectory, illustrating:

i) the observed averaged EMG activity

ii) the predicted EMG activity, as obtained by VBLS using data from all conditions

The third and fourth figures, Figures B.10(a) and B.11(a), show the observed and predicted velocities in the x and y directions, given M1 neural firing, while the hand is in the supinated wrist condition. These figures are for the (Kakei et al. 1999) M1 neural data set. As with the first two figures, the center plot shows the trajectories in eight different directions—in the (x, y) plane—taken by the hand. Each of the eight plots surrounding this center plot shows the velocities (in m/sec) over time for each hand trajectory, illustrating:

i) the observed velocities

ii) the predicted velocities, as obtained by VBLS using data from all conditions

Finally, the last two figures, Figures B.12(a) and B.13(a), show similar observed and predicted velocities in the x and y directions to Figures B.10(a) and B.11(a), except these correspond to neural firing in the PM cortex. The neural data set is from (Kakei et al. 2001).

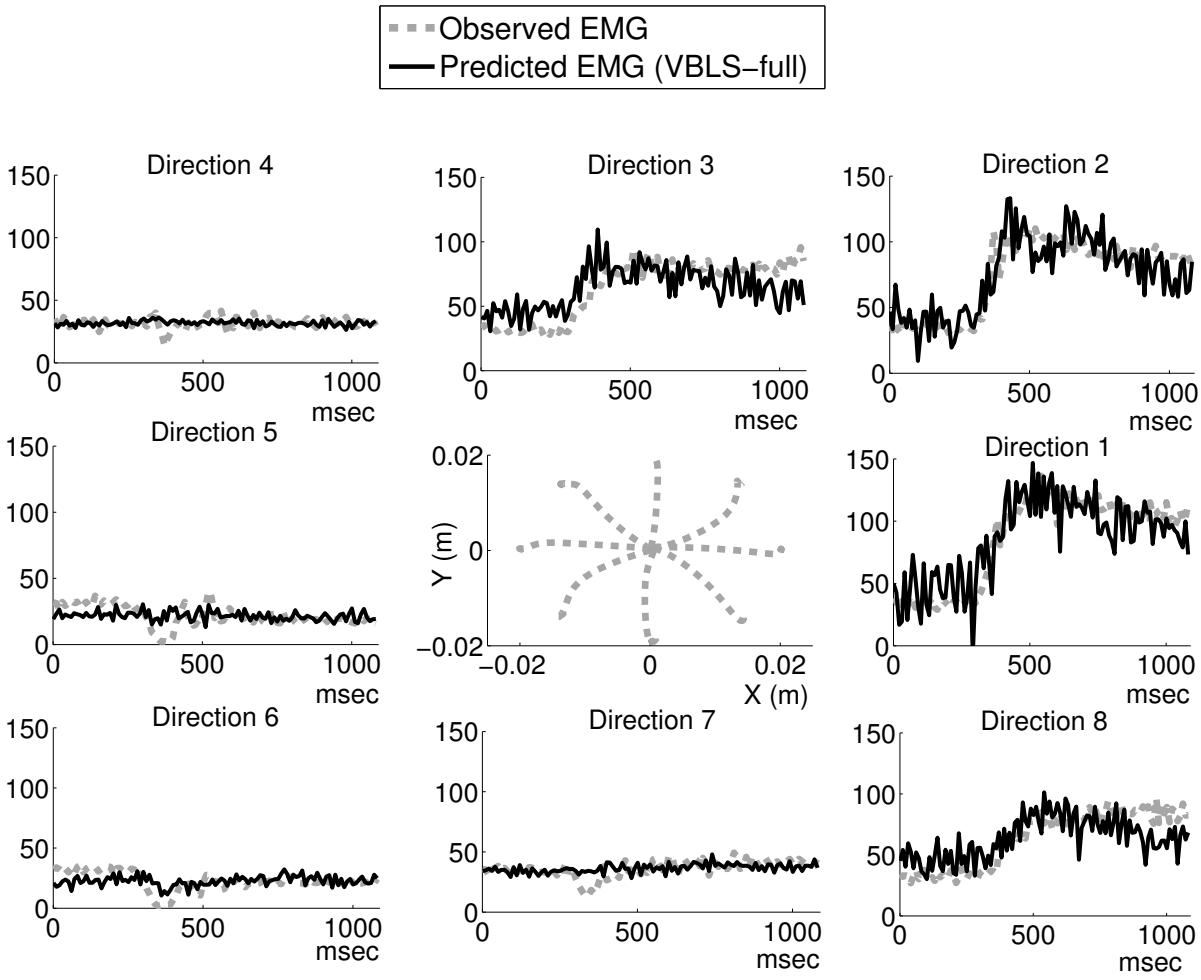
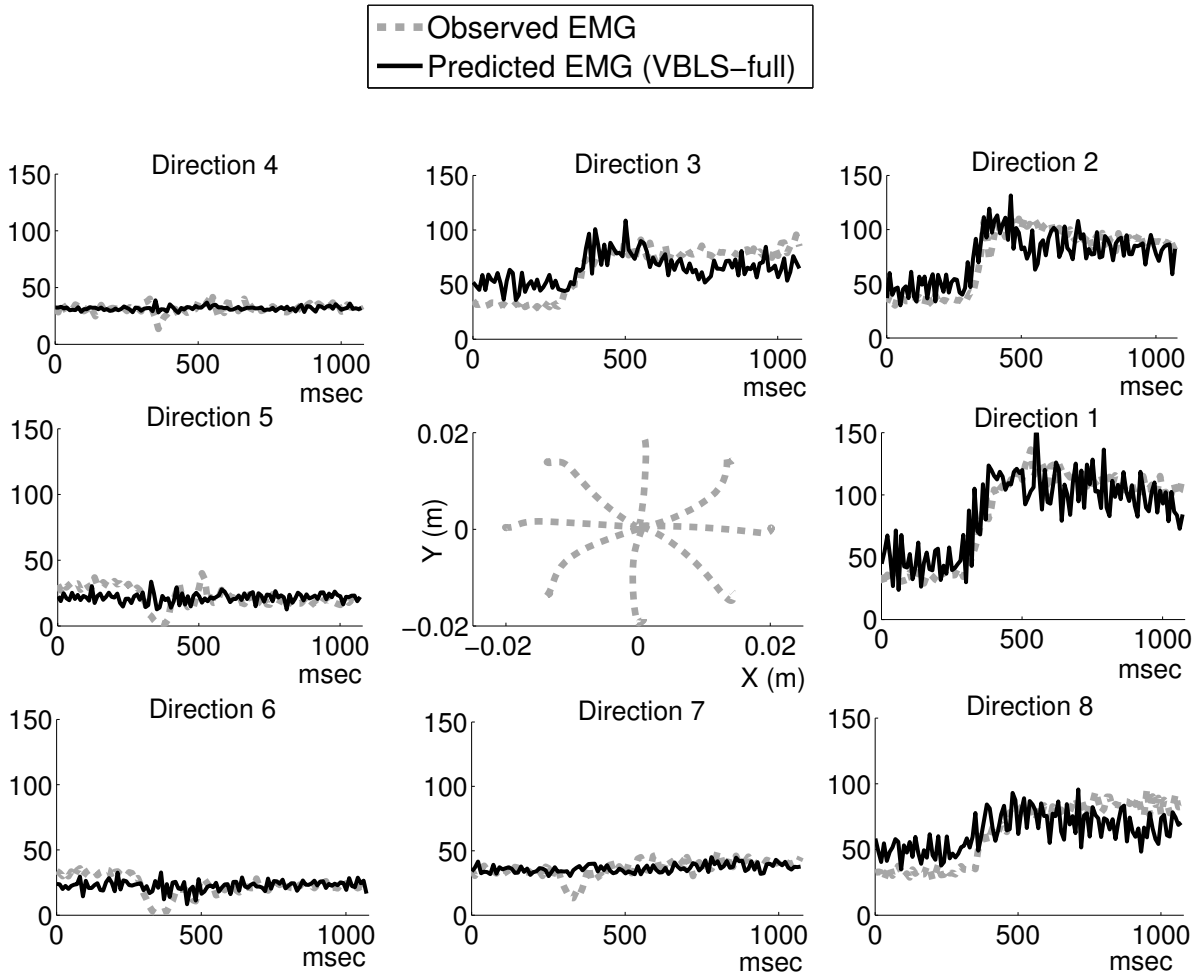


Figure B.8: Observed vs. predicted EMG traces for the ECRB muscle in the supinated wrist condition from the Kakei et al. (1999) M1 neural data set. VBLS-full is the result of applying VBLS on the entire data set.



(a) Observed vs. predicted EMG traces

Figure B.9: Observed vs. predicted EMG traces for the ECRB muscle in the supinated wrist condition from the Kakei et al. (2001) PM neural data set. VBLS-full is the result of applying VBLS on the entire data set.

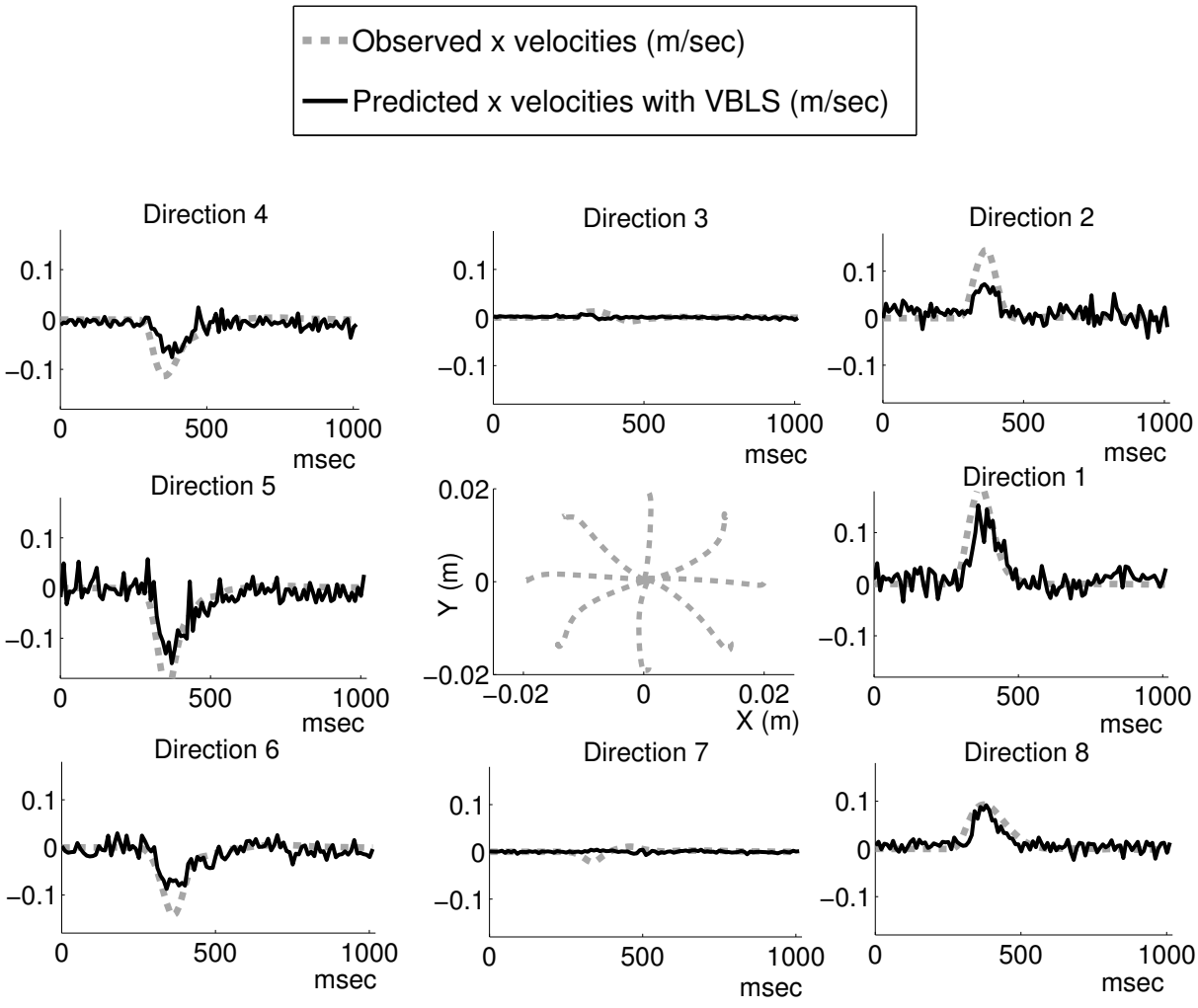


Figure B.10: Observed vs. predicted velocities in the x direction for the supinated wrist condition from the Kakei et al. (1999) M1 neural data set.

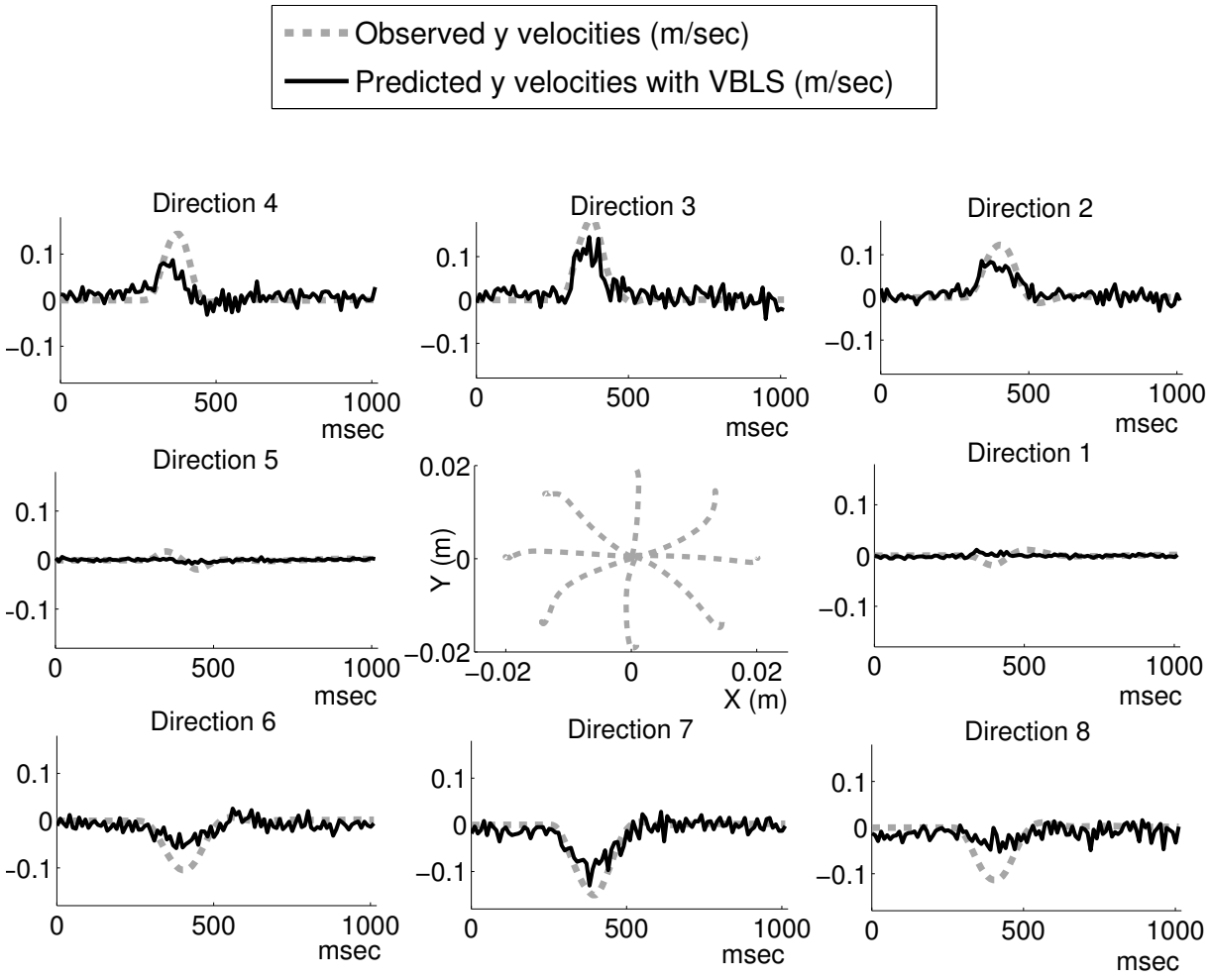


Figure B.11: Observed vs. predicted velocities in the y direction for the supinated wrist condition from the Kakei et al. (1999) M1 neural data set.

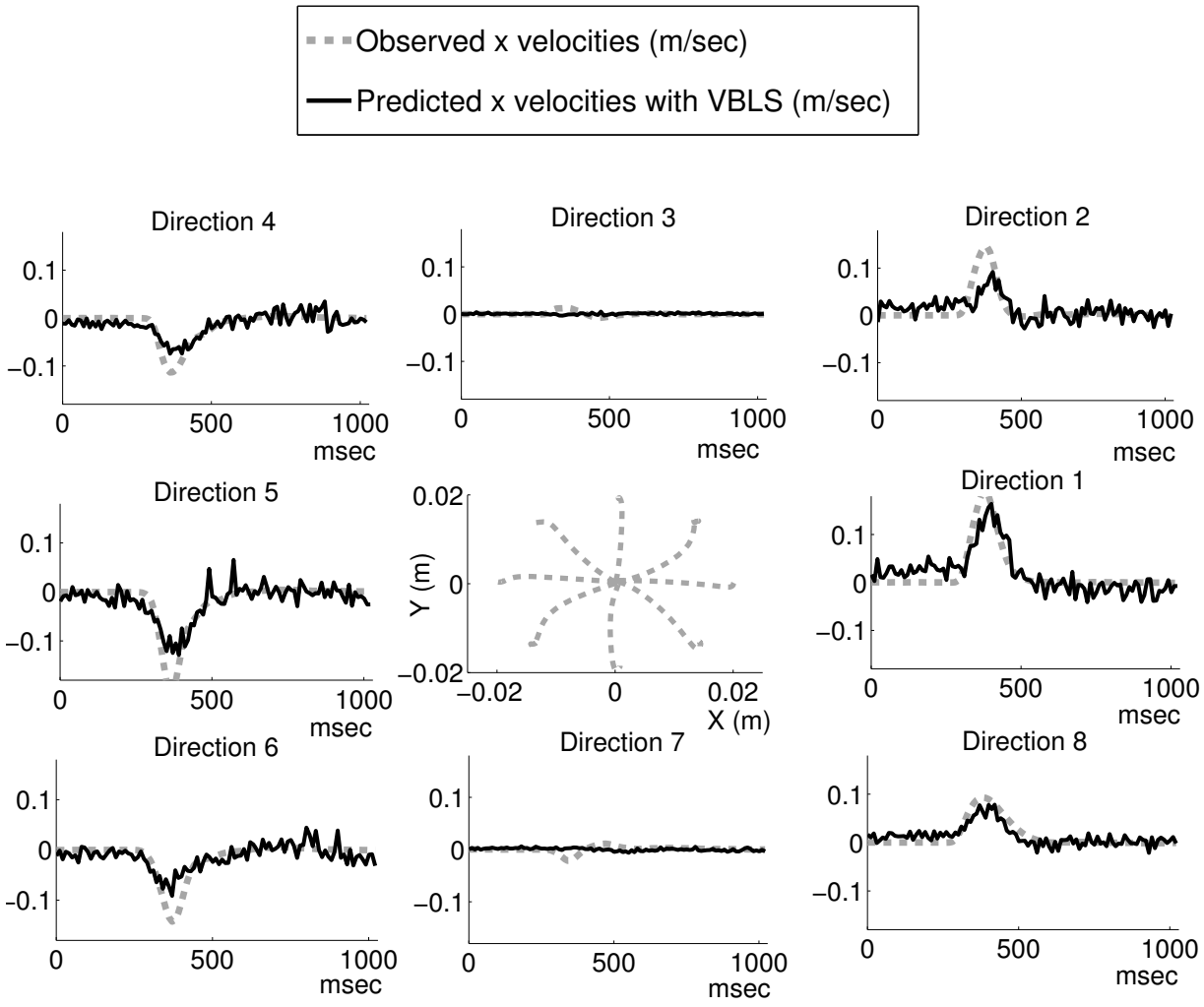


Figure B.12: Observed vs. predicted velocities in the x direction for the supinated wrist condition from the Kakei et al. (2001) PM neural data set.

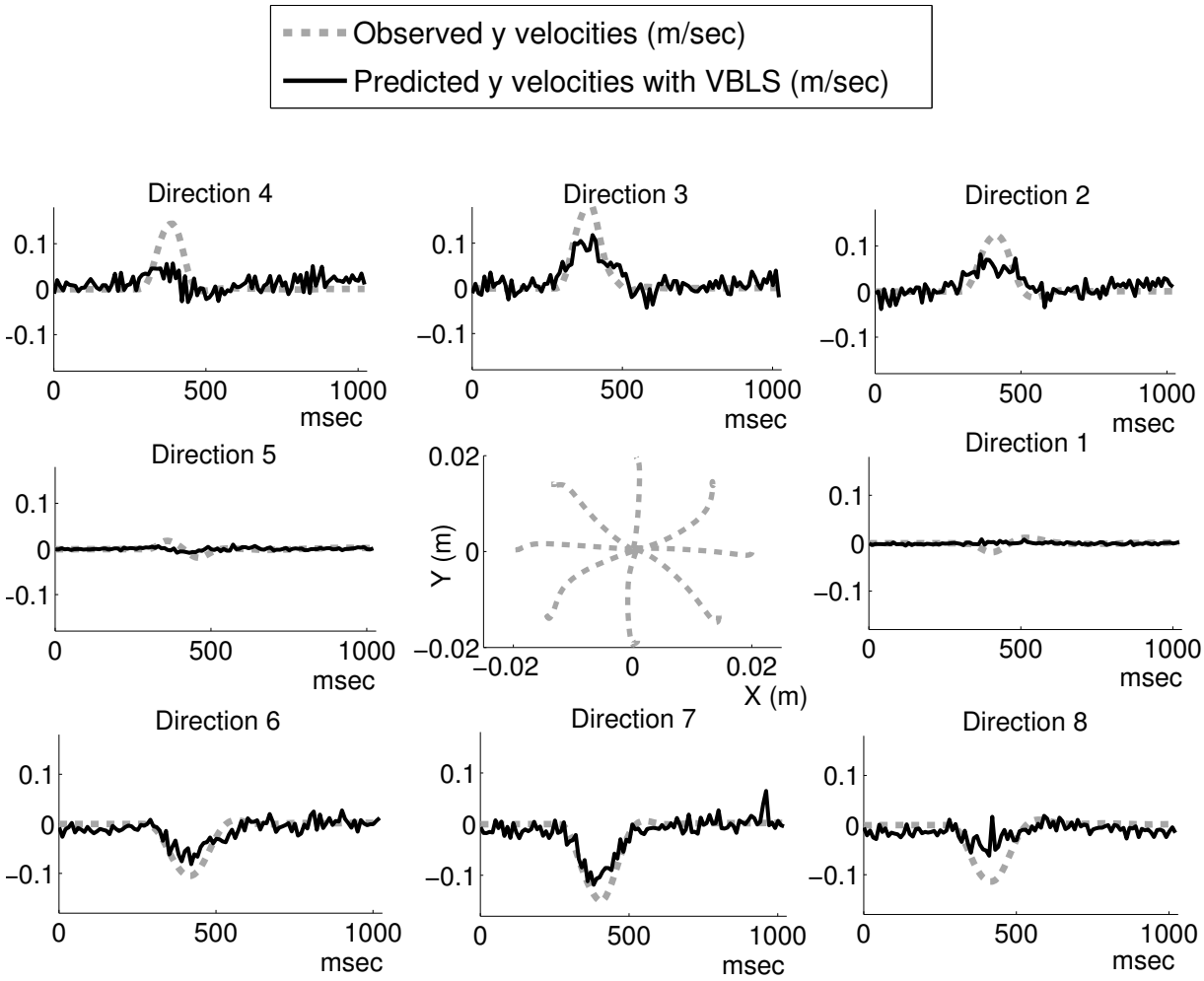


Figure B.13: Observed vs. predicted velocities in the y direction for the supinated wrist condition from the Kakei et al. (2001) PM neural data set.

Appendix C

Bayesian Factor Analysis for Regression

C.1 Factor Analysis for Regression

Factor Analysis, e.g., (Ghahramani & Hinton 1997), is a latent variable model used to detection structure in relationships between variables and to reduce the number of variables. The observed data $\mathbf{y} \in \Re^{p \times 1}$ is assumed to be generated from K latent variables or factors $\mathbf{t}$:

$$\mathbf{y}_i = \mathbf{W}\mathbf{t}_i + \epsilon_i$$

for all i, where $1 \leq i \leq N$ and N is the number of samples observed in the data set. The number of latent variables K is unknown and can be found, for example, by selecting the number of eigenvalues greater than 1. If the latent variables are assumed to be independently distributed as:

$$\begin{aligned} \mathbf{t}_i &\sim \text{Normal}\,(0, \mathbf{I}) \\ \epsilon_i &\sim \text{Normal}\,(0, \boldsymbol{\Psi})\,, \end{aligned}$$

then the factor loadings $\mathbf{W}$ and the diagonal noise variance matrix $\boldsymbol{\Psi}$ can be estimated using the EM algorithm (Ghahramani & Hinton 1997) or Bayesian techniques (Ghahramani & Beal 2000).

In Factor Analysis for regression (or joint-space Factor Analysis, since both $\mathbf{x}$ and y are jointly considered), we define a vector $\mathbf{z}$ that contains both the input data $\mathbf{x} \in \Re^{d \times 1}$ and scalar output data y:

$$\mathbf{z} \equiv \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} \quad \text{and} \quad \mathbf{W} \equiv \begin{bmatrix} \mathbf{W}_\mathbf{x} \\ \mathbf{W}_y \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Psi} \equiv \begin{bmatrix} \boldsymbol{\Psi}_\mathbf{x} & \mathbf{0} \\ \mathbf{0}^T & \psi_y \end{bmatrix}$$

where $\mathbf{W}_x$, $\mathbf{W}_y$ are factor loadings for $\mathbf{x}$ and y, respectively and $\boldsymbol{\Psi}_x$, ψ_y are the noise variances for $\mathbf{x}$ and y, respectively. Once we estimate $\mathbf{W}$ and $\boldsymbol{\Psi}$ for the joint data space of $\mathbf{z}$, we can condition y on $\mathbf{x}$ and marginalize out the latent variables $\mathbf{t}$. The joint distribution of the observed and latent variables will have the following mean and covariance:

$$\left\langle \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix} \right\rangle = 0$$

$$\mathrm{Cov}\left(\begin{bmatrix}\mathbf{x}\\ y\end{bmatrix}\right) = \begin{bmatrix}\mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi} & \mathbf{W}\\ \mathbf{W}^T & \mathbf{I}\end{bmatrix}$$

The covariance matrix can then be partitioned as follows:

$$\mathrm{Cov}\left(\begin{bmatrix}\mathbf{x}\\ y\end{bmatrix}\right) = \begin{bmatrix}\mathbf{W}_\mathbf{x}\mathbf{W}_\mathbf{x}^T + \boldsymbol{\Psi}_\mathbf{x} & \mathbf{W}_\mathbf{x}\mathbf{W}_y^T & \mathbf{W}_\mathbf{x}\\ \mathbf{W}_y^T\mathbf{W}_\mathbf{x}^T & \mathbf{W}_y\mathbf{W}_y^T + \psi_y & \mathbf{W}_y\\ \mathbf{W}_\mathbf{x}^T & \mathbf{W}_y^T & \mathbf{I}\end{bmatrix}$$

leading to the following conditional distribution:

$$\left\langle \begin{bmatrix}y\\ \mathbf{t}\end{bmatrix} | \mathbf{x} \right\rangle = \begin{bmatrix}\mathbf{W}_y\mathbf{W}_\mathbf{x}^T\\ \mathbf{W}_\mathbf{x}^T\end{bmatrix} \left(\mathbf{W}_\mathbf{x}\mathbf{W}_\mathbf{x}^T + \boldsymbol{\Psi}_\mathbf{x}\right)^{-1} \mathbf{x}$$

Marginalizing out $\mathbf{t}$, we obtain:

$$\begin{aligned}\langle y|\mathbf{x}\rangle &= \mathbf{W}_y\mathbf{W}_\mathbf{x}^T \left(\mathbf{W}_\mathbf{x}\mathbf{W}_\mathbf{x}^T + \boldsymbol{\Psi}_\mathbf{x}\right)^{-1} \mathbf{x}\\ &= \underbrace{\mathbf{W}_y \left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T\boldsymbol{\Psi}_\mathbf{x}^{-1}\mathbf{W}_\mathbf{x}\right)^{-1} \mathbf{W}_\mathbf{x}^T\boldsymbol{\Psi}_\mathbf{x}^{-1}}_{\mathbf{b}_{\mathrm{JFR}}^T} \mathbf{x}\end{aligned}$$

where $\langle\cdot\rangle$ indicates expectation and the matrix inversion lemma has been applied in the second line. Hence,

$$\mathbf{b}_{\mathrm{JFR}} = \boldsymbol{\Psi}_\mathbf{x}^{-1}\mathbf{W}_\mathbf{x} \left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T\boldsymbol{\Psi}_\mathbf{x}^{-1}\mathbf{W}_\mathbf{x}\right)^{-1} \mathbf{W}_y^T \tag{C.1}$$

Note that the required matrix inversion of $\left(\mathbf{I} + \mathbf{W}_\mathbf{x}^T\boldsymbol{\Psi}_\mathbf{x}^{-1}\mathbf{W}_\mathbf{x}\right)$ is of the order of the *latent* dimensionality K, which makes joint-space Factor Analysis for regression computationally attractive for problems in which the underlying latent variable manifold is known to be relatively low dimensional (i.e. $K \ll d$).

C.2 Derivation of Bayesian Joint Factor Analysis

To derive the final EM update equations for model in Section 3.2.2, we can proceed in a similar fashion as described in Appendix A.1. The complete log likelihood expression in Eq. (3.9) can be expanded to:

$$\begin{aligned}&\log p(\mathbf{y}, \mathbf{Z}, \mathbf{T}, \mathbf{w}_z, \mathbf{w}_x, \boldsymbol{\alpha}|\mathbf{X})\\ &= -\frac{N}{2}\log\psi_y - \frac{1}{2\psi_y}\sum_{i=1}^{N}\left(y_i - \mathbf{1}^T z_i\right)^2 - \frac{N}{2}\sum_{m=1}^{d}\log\psi_{zm} - \sum_{m=1}^{d}\frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\left(z_{im} - t_{im}w_{zm}\right)^2\\ &\quad - \frac{N}{2}\sum_{m=1}^{d}\log\psi_{xm} - \sum_{m=1}^{d}\frac{1}{2\psi_{xm}}\sum_{i=1}^{N}\left(x_{im} - t_{im}w_{xm}\right)^2\\ &\quad + \frac{1}{2}\sum_{m=1}^{d}\log\alpha_m - \frac{1}{2}\sum_{m=1}^{d}\alpha_m w_{zm}^2 + \frac{1}{2}\sum_{m=1}^{d}\log\alpha_m - \frac{1}{2}\sum_{m=1}^{d}\alpha_m w_{xm}^2\end{aligned}$$

$$-\frac{1}{2}\sum_{m=1}^{d}\sum_{i=1}^{N}t_{im}^2+\sum_{m=1}^{d}\left(a_{\alpha_{m,0}}-1\right)\log\alpha_m-\sum_{m=1}^{d}b_{\alpha_{m,0}}\alpha_m+\text{const}$$

The M-step equations can be derived by maximizing the complete log likelihood with respect to each of the parameters ψ_y, ψ_{zm} and ψ_{xm}.

Assuming that:

$$Q(\boldsymbol{\alpha},\mathbf{w}_z,\mathbf{w}_x,\mathbf{Z},\mathbf{T})=Q(\boldsymbol{\alpha})Q(\mathbf{w}_z)Q(\mathbf{w}_x)Q(\mathbf{Z},\mathbf{T}),$$

we can then infer the posterior distributions of the random variables (E-step updates) analytically:

- For $Q(\alpha)$:

$$\begin{aligned}\log Q(\alpha)&=\sum_{m=1}^{d}\Big\{\log\alpha_m-\frac{1}{2}\alpha_m\left\langle w_{zm}^2\right\rangle-\frac{1}{2}\alpha_m\left\langle w_{xm}^2\right\rangle\\&\quad+\left(a_{\alpha_{m,0}}-1\right)\log\alpha_m-b_{\alpha_{m,0}}\alpha_m\Big\}+\ \text{const}_\alpha\\&=\sum_{m=1}^{d}\left[a_{\alpha_{m,0}+1-1}\log\alpha_m-\left(b_{\alpha_{m,0}}+\frac{\left\langle w_{zm}^2\right\rangle+\left\langle w_{xm}^2\right\rangle}{2}\right)\alpha_m\right]+\ \text{const}_\alpha\end{aligned}$$

  Assuming $Q(\alpha)=\prod_{m=1}^{d}Q(\alpha_m)$ and $Q(\alpha_m)=\text{Gamma}(\hat{a}_{\alpha_m},\hat{b}_{\alpha_m})$,

$$\begin{aligned}\hat{a}_{\alpha_m}&=a_{\alpha_{m,0}}+1\\\hat{b}_{\alpha_m}&=b_{\alpha_{m,0}}+\frac{\left\langle w_{zm}^2\right\rangle+\left\langle w_{xm}^2\right\rangle}{2}\end{aligned}\tag{C.2}$$

- For $Q(\mathbf{W}_z)$:

$$\begin{aligned}\log Q(\mathbf{W}_z)&=\sum_{m=1}^{d}\left[-\frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\left\langle\left(z_{im}-w_{zm}t_{im}\right)^2\right\rangle-\frac{1}{2}\left\langle\alpha_m\right\rangle w_{zm}^2\right]+\ \text{const}_{\mathbf{W}_z}\\&=-\frac{1}{2}\sum_{m=1}^{d}\left[w_{zm}^2\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N}\left\langle t_{im}^2\right\rangle+\left\langle\alpha_m\right\rangle\right)-2w_{zm}\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N}\left\langle z_{im}t_{im}\right\rangle\right)\right]\\&\quad-\frac{1}{2}\left[\frac{1}{\psi_{zm}}\sum_{i=1}^{N}\left\langle z_{im}^2\right\rangle\right]+\ \text{const}_{\mathbf{W}_z}\end{aligned}$$

  Assuming $Q(\mathbf{W}_z)=\prod_{m=1}^{d}Q(w_{zm})$ and $Q(w_{zm})=\text{Normal}(\left\langle w_{zm}\right\rangle,\sigma_{w_{zm}}^2)$,

$$\begin{aligned}\sigma_{w_{zm}}^2&=\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N}\left\langle t_{im}^2\right\rangle+\left\langle\alpha_m\right\rangle\right)^{-1}\\\left\langle w_{zm}\right\rangle&=\sigma_{w_{zm}}^2\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N}\left\langle z_{im}t_{im}\right\rangle\right)\end{aligned}\tag{C.3}$$

- For $Q(\mathbf{W}_x)$:

$$\begin{aligned}\log Q(\mathbf{W}_x) &= \sum_{m=1}^{d}\left[-\frac{1}{2\psi_{xm}}\sum_{i=1}^{N}\left\langle(x_{im}-w_{xm}t_{im})^2\right\rangle-\frac{1}{2}\left\langle\alpha_m\right\rangle w_{xm}^2\right]+\ \text{const}_{\mathbf{W}_x}\\ &= -\frac{1}{2}\sum_{m=1}^{d}\left[w_{xm}^2\left(\frac{1}{\psi_{xm}}\sum_{i=1}^{N}\left\langle t_{im}^2\right\rangle+\left\langle\alpha_m\right\rangle\right)-2w_{xm}\left(\frac{1}{\psi_{zm}}\sum_{i=1}^{N}x_{im}\left\langle t_{im}\right\rangle\right)\right]\\ &\quad-\frac{1}{2}\left[\frac{1}{\psi_{xm}}\sum_{i=1}^{N}x_{im}^2\right]+\ \text{const}_{\mathbf{W}_y}\end{aligned}$$

Assuming $Q(\mathbf{W}_x)=\prod_{m=1}^{d}Q(w_{xm})$ and $Q(w_{xm})=\text{Normal}(\left\langle w_{xm}\right\rangle,\sigma_{w_{xm}}^2)$,

$$\begin{aligned}\sigma_{w_{xm}}^2 &= \left(\frac{1}{\psi_{xm}}\sum_{i=1}^{N}\left\langle t_{im}^2\right\rangle+\left\langle\alpha_m\right\rangle\right)^{-1}\\ \left\langle w_{xm}\right\rangle &= \sigma_{w_{xm}}^2\left(\frac{1}{\psi_{xm}}\sum_{i=1}^{N}x_{im}\left\langle t_{im}\right\rangle\right)\end{aligned}\tag{C.4}$$

- For $Q(\eta)$ (where $\eta = \{z_i, t_i\}$):

$$
\begin{aligned}
&\log Q(\eta) \\
&= -\frac{1}{2\psi_y}\sum_{i=1}^{N}\left\langle \left(y_i - \mathbf{1}^T z_i\right)^2 \right\rangle - \sum_{m=1}^{d}\frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\left\langle (z_{im} - t_{im}w_{zm})^2 \right\rangle \\
&\quad - \sum_{m=1}^{d}\frac{1}{2\psi_{xm}}\sum_{i=1}^{N}\left\langle (x_{im} - t_{im}w_{xm})^2 \right\rangle - \frac{1}{2}\sum_{m=1}^{d}\sum_{i=1}^{N}\left\langle t_{im}^2 \right\rangle \\
&= -\frac{1}{2}\sum_{i=1}^{N}\left[\frac{1}{\psi_y}\left(y_i - \mathbf{1}^T z_i\right)^2 + \left\langle (z_i - \mathbf{W}_y t_i)^T \Psi_z^{-1}(z_i - \mathbf{W}_y t_i)\right\rangle\right] \\
&\quad - \frac{1}{2}\sum_{i=1}^{N}\left\langle (x_i - \mathbf{W}_x t_i)^T \Psi_x^{-1}(x_i - \mathbf{W}_x t_i)\right\rangle - \frac{1}{2}\sum_{i=1}^{N}\left\langle t_i^T t_i\right\rangle \\
&= -\frac{1}{2}\sum_{i=1}^{N}\left[-2\frac{y_i \mathbf{1}^T z_i}{\psi_y} + z_i^T\frac{\mathbf{1}\mathbf{1}^T}{\psi_y}z_i + z_i^T\Psi_z^{-1}z_i - 2z_i^T\Psi_z^{-1}\left\langle \mathbf{W}_y\right\rangle t_i + t_i^T\left\langle \mathbf{W}_y^T\mathbf{W}_y\right\rangle\Psi_z^{-1}\right] \\
&\quad - \frac{1}{2}\sum_{i=1}^{N}\left[t_i^T\left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x\right\rangle\Psi_x^{-1}\right)t_i - 2t_i^T\left\langle \mathbf{W}_x\right\rangle^T\Psi_x^{-1}x_i\right] \\
&= -\frac{1}{2}\sum_{i=1}^{N}\left[z_i^T\left(\frac{\mathbf{1}\mathbf{1}^T}{\psi_y} + \Psi_z^{-1}\right)z_i + t_i^T\left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x\right\rangle\Psi_x^{-1} + \left\langle \mathbf{W}_y^T\mathbf{W}_y\right\rangle\Psi_z^{-1}\right)t_i\right] \\
&\quad - \frac{1}{2}\sum_{i=1}^{N}\left[-2\frac{y_i\sum_{m=1}^{d} z_{im}}{\psi_y} - 2\sum_{m=1}^{d}\frac{x_{im}\left\langle w_{xm}\right\rangle t_{im}}{\psi_{xm}} - 2\sum_{m=1}^{d}\frac{z_{im}\left\langle w_{zm}\alpha_m\right\rangle t_{im}}{\psi_{zm}}\right]
\end{aligned}
\tag{C.5}
$$

where $\left\langle \mathbf{W}_y^T\Psi_z^{-1}\mathbf{W}_y\right\rangle = \left\langle \mathbf{W}_y^T\mathbf{W}_y\right\rangle\Psi_z^{-1}$ since Ψ_z^{-1}, $\mathbf{W}_y$ are both diagonal matrices and, similarly, $\left\langle \mathbf{W}_x^T\Psi_x^{-1}\mathbf{W}_x\right\rangle = \left\langle \mathbf{W}_x^T\mathbf{W}_x\right\rangle\Psi_x^{-1}$.

Now, since we know the joint posterior of z and t is joint Gaussian distribution, it will take the form:

$$
\begin{aligned}
&\left(\begin{bmatrix} z_i \\ t_i \end{bmatrix} - \mu\right)^T \Sigma^{-1}\left(\begin{bmatrix} z_i \\ t_i \end{bmatrix} - \mu\right) \\
&= \begin{bmatrix} z_i \\ t_i \end{bmatrix}^T \Sigma^{-1}\begin{bmatrix} z_i \\ t_i \end{bmatrix} - 2\mu^T\Sigma^{-1}\begin{bmatrix} z_i \\ t_i \end{bmatrix} + \mu^T\mu
\end{aligned}
\tag{C.6}
$$

Expanding out the quadratic term in $[z_i\ t_i]^T$ from Eq. (C.6), we match up the coefficients of terms in Eq. (C.5) to find that the inverse posterior covariance matrix is:

$$
\Sigma^{-1} = \begin{bmatrix} A_{zz} & A_{zt} \\ A_{tz} & A_{tt} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{1}\mathbf{1}}{\psi_y} + \Psi_z^{-1} & -\Psi_z^{-1}\left\langle \mathbf{W}_y\right\rangle \\ -\left\langle \mathbf{W}_y\right\rangle^T\Psi_z^{-1} & \mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x\right\rangle\Psi_x^{-1} + \left\langle \mathbf{W}_y^T\mathbf{W}_y\right\rangle\Psi_z^{-1} \end{bmatrix}
\tag{C.7}
$$

To find Σ, we use the formula for the inverse of a partitioned matrix to get:

$$\Sigma = \begin{bmatrix} \Sigma_{zz} & \Sigma_{zt} \\ \Sigma_{tz} & \Sigma_{tt} \end{bmatrix} = \begin{bmatrix} \left(A_{zz} - A_{zt}A_{tt}^{-1}A_{tz}\right)^{-1} & -\Sigma_{zz}A_{zt}A_{tt}^{-1} \\ -A_{tt}^{-1}A_{zt}\Sigma_{zz} & A_{tt}^{-1} + A_{tt}^{-1}A_{zt}\Sigma_{zz}A_{zt}A_{tt}^{-1} \end{bmatrix}$$

We can use the Sherman Morrison Woodbury Theorem to simplify Σ_{zz}, noting that $\left\langle \mathbf{W}_y^T\mathbf{W}_y \right\rangle = \text{diag}\left(w_z w_z^T + \sigma_{w_z}^2\right) = \left(\Sigma_{\mathbf{W}_y}\right)_{mm} + \left\langle \mathbf{W}_y \right\rangle^T \left\langle \mathbf{W}_y \right\rangle$:

$$\begin{aligned}
\Sigma_{zz}^{-1} &= A_{zz} - A_{zt}A_{tt}^{-1}A_{tz} \\
&= \frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \Psi_z^{-1} \\
&\quad - \Psi_z^{-1}\left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left\langle \mathbf{W}_y^T\mathbf{W}_y \right\rangle \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_y \right\rangle^T \Psi_z^{-1} \\
&= \frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \Psi_z^{-1} \\
&\quad - \Psi_z^{-1}\left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left\langle \mathbf{W}_y^T\mathbf{W}_y \right\rangle \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_y \right\rangle^T \Psi_z^{-1} \\
&= \frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \Psi_z^{-1} \\
&\quad - \Psi_z^{-1}\left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + (\Sigma_{\mathbf{W}_y})_{mm}\Psi_x^{-1} + \left\langle \mathbf{W}_y \right\rangle^T \Psi_z^{-1} \left\langle \mathbf{W}_y \right\rangle\right)^{-1} \\
&\quad \left\langle \mathbf{W}_y \right\rangle^T \Psi_z^{-1} \\
&= \frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \left(\Psi_z + \left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left(\Sigma_{\mathbf{W}_y}\right)_{mm} \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_y \right\rangle^T\right)^{-1} \\
&\quad \text{leading to...} \\
\Sigma_{zz} &= \left[\frac{1}{\psi_y}\mathbf{1}\mathbf{1}^T + \left(\Psi_z + \left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left(\Sigma_{\mathbf{W}_y}\right)_{mm} \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_y \right\rangle^T\right)^{-1}\right]^{-1} \\
&= \mathbf{M} - \frac{\mathbf{M}\mathbf{1}\mathbf{1}^T\mathbf{M}}{\psi_y + \mathbf{1}^T\mathbf{M}\mathbf{1}} \text{ where...} \\
\mathbf{M} &= \Psi_z + \left\langle \mathbf{W}_y \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left(\Sigma_{\mathbf{W}_y}\right)_{mm} \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_y \right\rangle^T
\end{aligned}$$

Therefore, the full posterior covariance of z and t consists of the following block matrices:

$$\Sigma = \begin{bmatrix} \Sigma_{zz} & \Sigma_{zt} \\ \Sigma_{tz} & \Sigma tt \end{bmatrix}$$

where:

$$\begin{aligned}
\Sigma_{zz} &= \mathbf{M} - \frac{\mathbf{M}\mathbf{1}\mathbf{1}^T\mathbf{M}}{\psi_y + \mathbf{1}^T\mathbf{M}\mathbf{1}} \\
\Sigma_{zt} &= -\Sigma_{zz} \left\langle \mathbf{W}_z \right\rangle \Psi_z^{-1} \left(\mathbf{I} + \Psi_x^{-1} \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle + \left\langle \mathbf{W}_z^T\mathbf{W}_z \right\rangle \Psi_z^{-1}\right)^{-1} \\
\Sigma_{tz} &= \left(\Sigma_{zt}\right)^T \\
\Sigma_{tt} &= \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left\langle \mathbf{W}_z^T\mathbf{W}_z \right\rangle \Psi_z^{-1}\right)^{-1} + A_{tt}^{-1}A_{tz}\Sigma_{zz}A_{tz}A_{tt}^{-1} \\
\mathbf{M} &= \Psi_z + \left\langle \mathbf{W}_z \right\rangle \left(\mathbf{I} + \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle \Psi_x^{-1} + \left(\Sigma_{\mathbf{W}_z}\right)_{mm} \Psi_z^{-1}\right)^{-1} \left\langle \mathbf{W}_z \right\rangle^T
\end{aligned} \tag{C.8}$$

Notice that all the above equations involve diagonal matrices and, as a result, require a computational complexity that is linear in the number of dimensions.

To find the means of z_i and t_i, we refer to Eq. (C.6) and, in a similar manner for the covariance matrix, match up coefficients of the term linear in $[z_i\ t_i]^T$ to get:

$$\begin{aligned}
-2\mu\Sigma^{-1} \begin{bmatrix} z_i \\ t_i \end{bmatrix} &= -2 \left(\frac{y_i \sum_{m=1}^{d} z_{im}}{\psi_y} + \sum_{m=1}^{d} \frac{x_{im} \left\langle w_{xm} \right\rangle t_{im}}{\psi_{xm}} \right) \\
&= -2 \begin{bmatrix} \frac{y_i\mathbf{1}^T}{\psi_y} & x_i \left\langle W_x \right\rangle^T \Psi_x^{-1} \end{bmatrix} \begin{bmatrix} z_i \\ t_i \end{bmatrix}
\end{aligned}$$

so that:

$$\begin{aligned}
\left\langle z_i \right\rangle &= \begin{bmatrix} \frac{y_i\mathbf{1}^T}{\psi_y} & x_i \left\langle W_x \right\rangle^T \Psi_x^{-1} \end{bmatrix} \begin{bmatrix} \Sigma_{zz} \\ \Sigma_{tz} \end{bmatrix} \\
\left\langle t_i \right\rangle &= \begin{bmatrix} \frac{y_i\mathbf{1}^T}{\psi_y} & x_i \left\langle W_x \right\rangle^T \Psi_x^{-1} \end{bmatrix} \begin{bmatrix} \Sigma_{zt} \\ \Sigma_{tt} \end{bmatrix}
\end{aligned}$$

Therefore,

$$\begin{aligned}
\left\langle z_i \right\rangle &= \frac{y_i}{\psi_y}\mathbf{1}^T\Sigma_{zz} + x_i \left\langle \mathbf{W}_x \right\rangle^T \Psi_x^{-1}\Sigma_{tz} \\
\left\langle t_i \right\rangle &= -\frac{y_i}{\psi_y}\mathbf{1}^T\Sigma_{zz} \left\langle \mathbf{W}_z \right\rangle \Psi_z^{-1} \left(\mathbf{I} + \Psi_x^{-1} \left\langle \mathbf{W}_x^T\mathbf{W}_x \right\rangle + \left\langle \mathbf{W}_z^T\mathbf{W}_z \right\rangle \Psi_z^{-1}\right)^{-1} \\
&\quad + x_i \left\langle \mathbf{W}_x \right\rangle^T \Psi_x^{-1}\Sigma_{tt} \\
\sigma_z^2 &= \operatorname{diag}\left(\Sigma_{zz}\right) \\
\sigma_t^2 &= \operatorname{diag}\left(\Sigma_t\right) \\
\operatorname{cov}(z,t) &= \operatorname{diag}\left(\Sigma_{zt}\right)
\end{aligned} \tag{C.9}$$

As a final note:

$$\begin{aligned}
\left\langle w_{zm}^2 \right\rangle &= \left\langle w_{zm} \right\rangle^2 + \sigma_{w_{zm}}^2 & \left\langle w_{xm}^2 \right\rangle &= \left\langle w_{xm} \right\rangle^2 + \sigma_{w_{xm}}^2 \\
\left\langle t_{im}^2 \right\rangle &= \left\langle t_{im} \right\rangle^2 + \sigma_t^2 & \left\langle z_{im}^2 \right\rangle &= \left\langle z_{im} \right\rangle^2 + \sigma_z^2
\end{aligned}$$

C.3 Monitoring the Incomplete Log Likelihood

To know when to stop iterating through the EM algorithm above, we should monitor the incomplete log likelihood and stop when the value appears to have converged. To calculate the incomplete log likelihood, we need to integrate out the variables $\boldsymbol{\alpha}$, $\mathbf{w}_z$, $\mathbf{w}_x$, $\mathbf{Z}$, $\mathbf{T}$ from the complete log likelihood expression. But since the calculation of the true posterior distribution $Q(\theta)$ 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 can monitor the lower bound of the incomplete log likelihood instead.

In the derivation of the EM algorithm, we reached an estimate of $Q(\theta)$, where $\theta = \{\boldsymbol{\alpha}, \mathbf{w}_z, \mathbf{w}_x, \mathbf{Z}, \mathbf{T}\}$, to be $Q(\boldsymbol{\alpha})Q(\mathbf{w}_z)Q(\mathbf{w}_x)Q(\mathbf{Z}, \mathbf{T})$. The lower bound to the incomplete log likelihood, where $\phi = \{\psi_y, \psi_z, \psi_x\}$ is:

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

and this simplifies to:

$$\begin{aligned}
&\log p(\mathbf{y}|\mathbf{X};\phi) \geq \\
&\quad - \frac{N}{2} \log \psi_y - \frac{1}{2\psi_y} \sum_{i=1}^{N} \left(y_i - 2y_i \mathbf{1}^T \langle \mathbf{z}_i \rangle + \mathbf{1}^T \left\langle \mathbf{z}_i \mathbf{z}_i^T \right\rangle \mathbf{1} \right) \\
&\quad - \frac{N}{2} \sum_{m=1}^{d} \log \psi_{zm} - \sum_{m=1}^{d} \frac{1}{2\psi_{zm}} \sum_{i=1}^{N} \left(\left\langle z_{im}^2 \right\rangle - 2 \langle w_{zm} \rangle \langle z_{im} t_{im} \rangle + \left\langle w_{zm}^2 \right\rangle \left\langle t_{im}^2 \right\rangle \right) \\
&\quad - \frac{N}{2} \sum_{m=1}^{d} \log \psi_{xm} - \sum_{m=1}^{d} \frac{1}{2\psi_{xm}} \sum_{i=1}^{N} \left(x_{im}^2 - 2 \langle w_{xm} \rangle \langle t_{im} \rangle x_{im} + \left\langle w_{xm}^2 \right\rangle \left\langle t_{im}^2 \right\rangle \right) \\
&\quad - \frac{1}{2} \sum_{m=1}^{d} \sum_{i=1}^{N} \left\langle t_{im}^2 \right\rangle - \frac{1}{2} \sum_{m=1}^{d} \langle \alpha_m \rangle \left\langle w_{zm}^2 \right\rangle - \frac{1}{2} \sum_{m=1}^{d} \langle \alpha_m \rangle \left\langle w_{xm}^2 \right\rangle \\
&\quad - \sum_{m=1}^{d} a_{\alpha m 0} \log \hat{b}_{\alpha m} - \sum_{m=1}^{d} b_{\alpha m 0} \langle \alpha_m \rangle \\
&\quad - \frac{1}{2} \log |\boldsymbol{\Sigma}_{z,t}^{-1}| - \sum_{m=1}^{d} \log \hat{b}_{\alpha m} + \frac{1}{2} \sum_{m=1}^{d} \log \sigma_{w_{zm}}^2 + \frac{1}{2} \sum_{m=1}^{d} \log \sigma_{w_{xm}}^2 + \text{Constant}
\end{aligned} \tag{C.10}$$

Note that once the EM algorithm has converged and we have the final values for the unknown variables and point-estimated parameters, we still need to infer what the regression coefficient if we want to make predictions.

Appendix D

Bayesian Local Kernel Shaping

D.1 Deriving a Lower Bound Using Convex Duality

Eq. (5.8) contains a problematic term of:

$$-\log\left(1+(x_{im}-x_{qm})^{2r}\,h_m\right)$$

that prevents us from deriving an analytically tractable expression for the posterior of h_m. We can use a variational approach on concave/convex functions suggested by Jaakkola & Jordan (2000) to find analytically tractable expressions. Specifically, we can find a lower bound to the term using the convex duality approximations:

$$-\log\left(1+(x_{im}-x_{qm})^{2r}\,h_m\right)\geq\max_{\lambda_{im}}\left[-\lambda_{im}h_m\,(x_{im}-x_{qm})^{2r}-f^*(\lambda_{im})\right] \tag{D.1}$$

where λ_{im} is a variational parameter to be optimized and:

$$f^*(\lambda_{im})=\max_{\lambda_{im}}\left[-\lambda_{im}h_m\,(x_{im}-x_{qm})^{2r}+\log\left(1+(x_{im}-x_{qm})^{2r}\,h_m\right)\right]$$

We can then find the value of h_m that maximizes $f^*(\lambda_{im})$:

$$\begin{aligned}\frac{\partial f^*(\lambda_{im})}{\partial h_m}&=0\\ -(x_{im}-x_{qm})^{2r}\,\lambda_{im}+\frac{(x_{im}-x_{qm})^{2r}}{1+(x_{im}-x_{qm})^{2r}\,h_m}&=0\\ h_m\,(x_{im}-x_{qm})^{2r}&=\frac{1}{\lambda_{im}}-1\end{aligned}$$

Hence:

$$\begin{aligned}f^*(\lambda_{im})&=-\lambda_{im}\left(\frac{1}{\lambda_{im}}-1\right)+\log\left(1+\frac{1}{\lambda_{im}}-1\right)\\&=-1+\lambda_{im}-\log\lambda_{im}\end{aligned} \tag{D.2}$$

Substituting the expression for $f^*(\lambda_{im})$ in Eq. (D.1) into the lower bound Eq. (D.2), we get:

$$-\lambda_{im} h_m \left(x_{im} - x_{qm}\right)^{2r} - f^*(\lambda_{im}) = -\lambda_{im} h_m \left(x_{im} - x_{qm}\right)^{2r} + 1 - \lambda_{im} + \log \lambda_{im} \quad \text{(D.3)}$$

and find the optimal value of the variational parameter λ_{im} by maximizing Eq. (D.3):

$$\begin{aligned}
\frac{\partial}{\partial \lambda_{im}} \left[-\lambda_{im} \left\langle h_m \right\rangle \left(x_{im} - x_{qm}\right)^{2r} + 1 - \lambda_{im} + \log \lambda_{im}\right] &= 0 \\
- \left\langle h_m \right\rangle \left(x_{im} - x_{qm}\right)^{2r} - 1 + \frac{1}{\lambda_{im}} &= 0 \\
\frac{1}{\lambda_{im}} &= 1 + \left\langle h_m \right\rangle \left(x_{im} - x_{qm}\right)^{2r} \\
\lambda_{im} &= \frac{1}{1 + \left\langle h_m \right\rangle \left(x_{im} - x_{qm}\right)^{2r}}
\end{aligned}$$

In summary, we can find a lower bound to the problem log term in Eq. (5.8) in the form of:

$$-\log\left(1 + \left(x_{im} - x_{qm}\right)^{2r} h_m\right) \geq -\lambda_{im} h_m \left(x_{im} - x_{qm}\right)^{2r} + 1 - \lambda_{im} + \log \lambda_{im} \quad \text{(D.4)}$$

where the lower bound is maximized when:

$$\lambda_{im} = \frac{1}{1 + \left\langle h_m \right\rangle \left(x_{im} - x_{qm}\right)^{2r}} \quad \text{(D.5)}$$

D.2 Derivation of Bayesian Kernel Shaping

To derive the final EM update equations for the model in Section 5.1.1, we can proceed in a similar fashion as described in Appendix A.1. Assuming that:

$$Q(\mathbf{b}, \psi_z, \mathbf{h}, \mathbf{z}) = Q(\mathbf{b}, \psi_z) Q(\mathbf{h}) Q(\mathbf{z})$$

we can infer the posterior distributions of the random variables (E-step updates) analytically:

- For $Q(\mathbf{b}|\psi_z)$:
  Assuming that $Q(\mathbf{b}) = \prod_{m=1}^{d} Q(\mathbf{b}_m)$:

$$\begin{aligned}
&\log Q(\mathbf{b}_m|\psi_{zm}) \\
&= -\frac{1}{2\psi_{zm}} \sum_{i=1}^{N} \left\langle w_i \right\rangle \left\langle \left(z_{im} - \mathbf{b}_m^T \mathbf{x}_{im}\right)^2 \right\rangle - \frac{1}{2\psi_{zm}} \mathbf{b}_m^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \mathbf{b}_m + \text{const}_{\mathbf{b}_m|\psi_{zm}} \\
&= -\frac{1}{2\psi_{zm}} \sum_{i=1}^{N} \left\langle w_i \right\rangle \left(\left\langle z_{im}^2 \right\rangle - 2 \left\langle z_{im} \right\rangle \left\langle \mathbf{b}_m \right\rangle^T \mathbf{x}_{im} + \mathbf{b}_m^T \mathbf{x}_{im} \mathbf{x}_{im}^T \mathbf{b}_m\right) \\
&\quad - \frac{1}{2\psi_{zm}} \mathbf{b}_m^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \mathbf{b}_m + \text{const}_{\mathbf{b}_m|\psi_{zm}}
\end{aligned}$$

$$= -\frac{1}{2}\left[\mathbf{b}_m^T \frac{1}{\psi_{zm}}\left(\boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} + \sum_{i=1}^{N}\langle w_i\rangle \mathbf{x}_{im}\mathbf{x}_{im}^T\right)\mathbf{b}_m - 2\mathbf{b}_m^T \frac{1}{\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle \langle z_{im}\rangle \mathbf{x}_{im}\right] + \text{const}_{\mathbf{b}_m|\psi_{zm}}$$

then the posterior covariance and mean of $\mathbf{b}_m$ is:

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{b}_m} &= \left(\boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} + \frac{1}{\psi_{zmN}}\sum_{i=1}^{N}\langle w_i\rangle \mathbf{x}_{im}\mathbf{x}_{im}^T\right)^{-1} \\ \langle \mathbf{b}_m\rangle &= \boldsymbol{\Sigma}_{\mathbf{b}_m}\left(\sum_{i=1}^{N}\langle w_i\rangle \langle z_{im}\rangle \mathbf{x}_{im}\right)\end{aligned} \tag{D.6}$$

- For $Q(\psi_z)$:
  Assuming $Q(\psi_z) = \prod_{m=1}^{d} Q(\psi_{zm})$,

$$\begin{aligned}&\log Q(\psi_{zm}) \\ &= -\frac{1}{2}\log\psi_{zm}\sum_{i=1}^{N}\langle w_i\rangle - \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle\left\langle\left(z_{im} - \mathbf{b}_m^T\mathbf{x}_{im}\right)^2\right\rangle - \frac{1}{2}\log\psi_{zm} \\ &\quad - \frac{1}{2\psi_{zm}}\left\langle \mathbf{b}_m T\boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1}\mathbf{b}_m\right\rangle - \left(\frac{n_{m0}}{2}+1\right)\log\psi_{zm} - \frac{n_{m0}}{2}\frac{\psi_{zmN0}}{\psi_{zm}} \\ &\quad - \int Q(\mathbf{b}_m|\psi_{zm})\log Q(\mathbf{b}_m|\psi_{zm})d\mathbf{b}_m + \text{const}_{\psi_{zm}} \\ &= -\log\psi_{zm}\left(\frac{\sum_{i=1}^{N}\langle w_i\rangle + n_{m0} + 1}{2} + 1\right) \\ &\quad - \frac{1}{2\psi_{zm}}\left(n_{m0}\psi_{zmN0} + \left\langle \mathbf{b}_m^T\boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1}\mathbf{b}_m\right\rangle + \sum_{i=1}^{N}\langle w_i\rangle\left\langle\left(z_{im} - \mathbf{b}_m^T\mathbf{x}_{im}\right)^2\right\rangle\right) \\ &\quad + \frac{1}{2}\left\{\log(2\pi\psi_{zm}\boldsymbol{\Sigma}_{\mathbf{b}_m}) + 1\right\} + \text{const}_{\psi_{zm}}\end{aligned}$$

  Given that:

$$\begin{aligned}&\frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle\left\langle\left(z_{im} - \mathbf{b}_m^T\mathbf{x}_{im}\right)^2\right\rangle \\ &= \frac{1}{2\psi_{zm}}\left[\sum_{i=1}^{N}\langle w_i\rangle\left\langle z_{im}^2\right\rangle - 2\sum_{i=1}^{N}\langle w_i\rangle z_{im}\mathbf{x}_{im}^T\langle\mathbf{b}_m\rangle + \sum_{i=1}^{N}\langle w_i\rangle \mathbf{x}_{im}^T\left\langle\mathbf{b}_m\mathbf{b}_m^T\right\rangle\mathbf{x}_{im}\right] \\ &= \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle\left(\langle z_{im}\rangle - \mathbf{x}_{im}^T\langle\mathbf{b}_m\rangle\right)^2 + \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle \mathbf{x}_{im}^T\psi_{zm}\boldsymbol{\Sigma}_{\mathbf{b}_m}\mathbf{x}_{im} \\ &\quad + \frac{1}{2\psi_{zm}}\sum_{i=1}^{N}\langle w_i\rangle \boldsymbol{\Sigma}_{\mathbf{z}_i|\mathbf{y}_i,\mathbf{x}_i}\end{aligned}$$

and

$$-\frac{1}{2\psi_{zm}}\left\langle \mathbf{b}_m^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \mathbf{b}_m \right\rangle = -\frac{1}{2\psi_{zm}} \langle \mathbf{b}_m \rangle^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \langle \mathbf{b}_m \rangle$$

we then get:

$$\begin{aligned}
&\log Q(\psi_{zm}) \\
&= \log \psi_{zm} \left(n_{m0} + \sum_{i=1}^{N} \langle w_i \rangle \right) - \frac{1}{2\psi_{zm}} \Big[n_{m0}\psi_{zmN0} + \langle \mathbf{b}_m \rangle^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \langle \mathbf{b}_m \rangle \\
&\quad + \sum_{i=1}^{N} \langle w_i \rangle \left(\langle z_{im} \rangle - \mathbf{x}_{im}^T \langle \mathbf{b}_m \rangle \right)^2 + \sum_{i=1}^{N} \langle w_i \rangle \boldsymbol{\Sigma}_{\mathbf{z}_i|\mathbf{y}_i,\mathbf{x}_i} \Big] + \text{const}_{\psi_{zm}}
\end{aligned}$$

giving us:

$$\begin{aligned}
n_m &= n_{m0} + \sum_{i=1}^{N} \langle w_i \rangle \\
\psi_{zmN} &= \frac{1}{n_0 + \sum_{i=1}^{N} \langle w_i \rangle} \Big[n_{m0}\psi_{zmN0} + \langle \mathbf{b}_m \rangle^T \boldsymbol{\Sigma}_{\mathbf{b}_m,0}^{-1} \langle \mathbf{b}_m \rangle \\
&\qquad + \sum_{i=1}^{N} \langle w_i \rangle \left(\langle z_{im} \rangle - \mathbf{x}_{im}^T \langle \mathbf{b}_m \rangle \right)^2 + \sum_{i=1}^{N} \langle w_i \rangle \boldsymbol{\Sigma}_{\mathbf{z}_i|\mathbf{y}_i,\mathbf{x}_i} \Big]
\end{aligned} \tag{D.7}$$

- For $Q(\mathbf{h})$:

  Assuming $Q(\mathbf{h}) = \prod_{m=1}^{d} Q(h_m)$:

$$\begin{aligned}
\log Q(\mathbf{h}) &= \sum_{m=1}^{d} \sum_{i=1}^{N} (1 - \langle w_i \rangle) \log h_m \left(x_{im} - x_{qm} \right)^{2r} - \sum_{m=1}^{d} \sum_{i=1}^{N} \lambda_i \left(x_{im} - x_{qm} \right)^{2r} h_m \\
&\quad + \sum_{m=1}^{d} (a_{hm0} - 1) \log h_m - \sum_{m=1}^{d} b_{hm0} h_m + \text{const}_{\mathbf{h}} \\
&= \sum_{m=1}^{d} \left(a_{hm0} + N - \sum_{i=1}^{N} \langle w_i \rangle - 1 \right) \log h_m \\
&\quad - \sum_{m=1}^{d} \left(b_{hm0} + \sum_{i=1}^{N} \lambda_{im} \left(x_{im} - x_{qm} \right)^{2r} \right) h_m + \text{const}_{\mathbf{h}}
\end{aligned}$$

  where $\lambda_{im} = \frac{1}{1 + \langle h_m \rangle (x_{im} - x_{qm})^{2r}}$. From the above, we can infer the posterior mean of h_m to be:

$$\langle h_m \rangle = \frac{a_{hm0} + N - \sum_{i=1}^{N} \langle w_i \rangle}{b_{hm0} + \sum_{i=1}^{N} \lambda_{im} \left(x_{im} - x_{qm} \right)^{2r}} \tag{D.8}$$

- For $Q(\mathbf{z}_i)$:

  Inference of the posterior distribution of $\mathbf{z}_i$ is done in a similar manner as in Appendix A.1.