DESIGN AND ANALYSIS OF OPTIMAL DECODING MODELS FOR BRAIN-MACHINE INTERFACES

By

SUNG-PHIL KIM

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2005
Copyright 2005

by

Sung-Phil Kim
This document is dedicated to my mother and my wife.
ACKNOWLEDGMENTS

I would like to thank God for His endless love to give me the best all the time in my life. I would also like to thank my mother, brothers and my wife for their love, supports, and solid belief in me.

I would like to thank Dr. Jose C. Principe for his untiring support, advice, and guidance. I will never forget his inspiration for me to make me think as a researcher.

I am very grateful to Dr. John G. Harris, Dr. Michael C. Nechyba, Dr. Karl Gugel, and Dr. Mark C.K.Yang for their support and advice for the brain-machine interfaces research.

I am also exceptionally grateful to Dr. Justin C. Sanchez, Dr. Yadunandana N. Rao, Dr. Deniz Ergodmus, and Shalom Darmanjian for their sincere support and collaboration. I must acknowledge Dr. Miguel A.L. Nicolelis and Dr. Jose M. Carmena for the opportunity to conduct this research with their support.

Final thanks go to Yuseok Ko, Jeongho Cho and Dongho Han, who have always helped and encouraged me.
TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................................................................ iv

LIST OF TABLES ........................................................................................................................................ viii

LIST OF FIGURES ........................................................................................................................................ ix

ABSTRACT ................................................................................................................................................... xiii

CHAPTER

1 INTRODUCTION ........................................................................................................................................ 1

Review of BMI Signal Processing ............................................................................................................. 3
Approaches .................................................................................................................................................. 6
Outline ........................................................................................................................................................ 8

2 EXPERIMENTAL SETUPS FOR BRAIN-MACHINE INTERFACES ................................................. 10

Recording of Electrical Activity of Neuronal Ensembles ........................................................................ 10
Behavioral Tasks ........................................................................................................................................ 11
Properties of Data ........................................................................................................................................ 12
Neuronal Firing Patterns ........................................................................................................................... 13
Hand Movements....................................................................................................................................... 16

3 LINEAR MODELING ............................................................................................................................... 19

Linear Modeling for BMIs......................................................................................................................... 19
The Wiener Filter ......................................................................................................................................... 23
Stochastic Gradient Learning .................................................................................................................... 27
Other Linear Modeling .............................................................................................................................. 28

4 REGULARIZED LINEAR MODELING ................................................................................................. 31

Dimension Reduction Using Subspace Projection ...................................................................................... 31
A Hybrid Subspace Projection .................................................................................................................... 32
Design of a Decoding Model Using the Subspace Wiener Filter ............................................................... 35
Parsimonious Modeling in Time Using the Gamma Filter ...................................................................... 37
Regularization by Parameter Constraints ................................................................................................. 41
Review of Shrinkage Methods .................................................................43
  Shrinkage methods ..............................................................................43
  The relationship between subspace projection and ridge regression ...45
  Comparison of shrinkage methods .......................................................45
  Regularization Based on the $L_2$-Norm Penalty ..................................47
  Regularization Based on the $L_1$-norm Penalty ....................................50

5 NONLINEAR MIXTURE OF MULTIPLE LINEAR MODELS ....................54
  Nonlinear Mixture of Linear Models Approach ....................................55
  Nonlinear Mixture of Competitive Linear Models ...............................55
  Time Delay Neural Networks ..............................................................59
  BMIs Design Using NMCLM ...............................................................59
  Analysis ..............................................................................................60
    Evaluation of Training Performance for NMCLM ..............................60
    Analysis of Linear Filters ..................................................................62

6 COMPARISON OF MODELS .................................................................64
  Comparison of Model Parameters .......................................................67
  Performance Evaluation .......................................................................69
  Statistical Performance Comparison ..................................................70

7 MULTIRESOLUTION ANALYSIS FOR BMI ...........................................73
  Multiresolution Analysis of Neuronal Spike Trains ...............................76
    Multiresolution Analysis ....................................................................77
    Multiresolution Analysis for the BMI Data .........................................80
    The Analysis of the Linear Model Based on the Multiresolution
      Representation ..................................................................................83
  Comparison of Models with the Multiresolution Representation ..........85
  Combination of Linear and Nonlinear Models .......................................89
    Nonlinear Modeling ..........................................................................91
    Simulations ......................................................................................93
  Discussions ..........................................................................................95

8 DETERMINATION OF NEURONAL FIRING PATTERNS USING NON-
  NEGATIVE MATRIX FACTORIZATION ..................................................99
  Nonnegative Matrix Factorization .......................................................101
  Factorization of Neuronal Bin Count Matrix .......................................103
    Data Preparation ................................................................................103
      3D food reaching data ....................................................................103
      2D target reaching data ..................................................................105
    Analysis of Factorization Process ....................................................105
      Choice of the number of bases .......................................................106
      How does NMF find repeated patterns? .........................................106
      Local minima problem ...................................................................110
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>The distributions of the sorted neuronal activity for each monkey in motor cortical areas</td>
</tr>
<tr>
<td>4-1</td>
<td>Procedure of the LAR algorithm</td>
</tr>
<tr>
<td>6-1</td>
<td>The generalization performances of linear models and nonlinear models for the 3D food reaching task</td>
</tr>
<tr>
<td>6-2</td>
<td>The generalization performances of linear models and nonlinear models for the 2D target reaching task</td>
</tr>
<tr>
<td>6-3</td>
<td>The t-test results for the difference of the magnitude of error vectors from the test dataset between the Wiener filter and other models</td>
</tr>
<tr>
<td>7-1</td>
<td>The number of the selected neurons in each cortical area</td>
</tr>
<tr>
<td>7-2</td>
<td>The number of the nonzero weights</td>
</tr>
<tr>
<td>7-3</td>
<td>The number of neurons selected by LAR for each models</td>
</tr>
<tr>
<td>7-4</td>
<td>Performance comparison between the multiresolution and the single resolution models</td>
</tr>
<tr>
<td>7-5</td>
<td>Performance comparison between the combinatory model and the single linear model</td>
</tr>
<tr>
<td>8-1</td>
<td>Comparison of important neurons; food reaching</td>
</tr>
<tr>
<td>8-2</td>
<td>Comparison of important neurons: target reaching</td>
</tr>
<tr>
<td>8-3</td>
<td>Performance evaluation of the Wiener filter and the mixture of multiple models based on NMF</td>
</tr>
<tr>
<td>9-1</td>
<td>Procedure of the LAR algorithm: revisited</td>
</tr>
<tr>
<td>9-2</td>
<td>The modified LAR algorithm for on-line variable selection</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>A system identification block diagram for BMIs.</td>
<td>2</td>
</tr>
<tr>
<td>2-1</td>
<td>An experimental setup of 3D reaching task.</td>
<td>12</td>
</tr>
<tr>
<td>2-2</td>
<td>An experimental setup of 2D target reaching task. The monkey moves a cursor (yellow circle) to a randomly placed target (green circle), and rewarded if a cursor intersects the target.</td>
<td>13</td>
</tr>
<tr>
<td>2-3</td>
<td>An example of the binned data.</td>
<td>13</td>
</tr>
<tr>
<td>2-4</td>
<td>The plots of the average (dot) and the standard deviation (bar) for each neuron of three monkeys.</td>
<td>14</td>
</tr>
<tr>
<td>2-5</td>
<td>The trajectories of the estimated mean firing rates for movement (solid line) and rest (dotted line) over sequence of subsets.</td>
<td>15</td>
</tr>
<tr>
<td>2-6</td>
<td>Illustrations of nonstationary properties of the input autocorrelation matrix.</td>
<td>16</td>
</tr>
<tr>
<td>2-7</td>
<td>Sample trajectories of (a) 3D food reaching, and (b) 2D target reaching movements.</td>
<td>17</td>
</tr>
<tr>
<td>2-8</td>
<td>The db6 continuous wavelet coefficients of trajectory signals of (a) 3D food reaching, and (b) 2D target reaching.</td>
<td>18</td>
</tr>
<tr>
<td>3-1</td>
<td>The topology of the linear filter designed for BMIs in the case of the 3D reaching task.</td>
<td>20</td>
</tr>
<tr>
<td>3-2</td>
<td>The Hinton diagram of the weights of the Wiener filter for food reaching.</td>
<td>26</td>
</tr>
<tr>
<td>3-3</td>
<td>The Hinton diagram of the weights of the Wiener filter for target reaching.</td>
<td>27</td>
</tr>
<tr>
<td>4-1</td>
<td>The overall diagram of the subspace Wiener filter.</td>
<td>34</td>
</tr>
<tr>
<td>4-2</td>
<td>The contour map of the validation MSE for (a) food reaching, and (b) target reaching.</td>
<td>35</td>
</tr>
<tr>
<td>4-3</td>
<td>The first three projection vectors in PCA for (a) food reaching, and (c) target reaching, and PLS for (b) food reaching, and (d) target reaching, respectively.</td>
<td>37</td>
</tr>
</tbody>
</table>
7-4 The distribution of the selected input variables for (a) x-coordinate, (b) and y-coordinate of position, and (c) x-coordinate, and (d) y-coordinate of velocity...86

7-5 The CEM curves of the single resolution model (red dotted lines), and the multiresolution model (black solid lines).................................................................89

7-6 An example of the residual trajectory from a linear model (the x-coordinate)......93

7-7 An example of the output trajectories of the combinatory network and single linear model........................................................................................................................95

7-8 Tap outputs from two generalized feedforward filters for a neuronal bin count input with different delay: the gamma, and Haar wavelet..................................................97

8-1 Segmentation of the reaching trajectories: reach from rest to food, reach from food to mouth, and reach from mouth to rest position ................................................104

8-2 The NMF results for food reaching........................................................................111

8-3 The NMF results for target reaching ....................................................................114

8-4 The hand position samples collected along with peaks in each NMF encoding (left), and the mean and variance of each set (right)......................................................116

8-5 The probabilities of the occurrence for hand position to be in each of sixteen angle bins. ................................................................................................................117

8-6. Tuning curve of neuronal firing patterns encoded in each NMF basis for 16 angle bins. ................................................................................................................118

9-1 The diagram of the architecture of real time neuronal subset selection method....131

9-2 An illustration of the successive maximum correlation over stages in the case of two variables (channels)..............................................................................134

9-3 Examples of the maximum absolute correlation curve in LAR.........................135

9-4 Neuronal subset selection examples.....................................................................137

9-5 Demonstration of filter outputs before subset selection; (top) synchronized data, (bottom) de-synchronized data.................................................................139

9-6 Neuronal subset selection conditioned by the correlation between filter outputs and desired response. ..................................................................................142

9-7 Demonstration of the robustness of the algorithm to initial conditions. ..........143

9-8 An example of the outputs of two tracking systems with (solid line), and without on-line channel selection (dashed line).................................................................144
9-9 Neuronal subset selection for all three coordinates of food reaching movement. 145

9-10 Neuronal subset selection over 2,000-second data; (a) subsets in the early part, and (b) subsets in the late part of the data. ................................................................. 146

9-11 Neuronal subset selection for a 2D target reaching BMI. ....................................................... 148

9-12 2D hand trajectories in five sample data segments selected in Fig. 9-11............... 148

9-13 Selection of individual neurons over a series of reaching movements. ................. 151

9-14 The distribution of the subset size over a series of reaching movements. ............ 153

9-15 Comparison of the average misadjustment per movement between the standard MIMO system learned by LMS and the MIMO system with on-line channel selection ................................................................. 153
Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

DESIGN AND ANALYSIS OF OPTIMAL DECODING MODELS FOR BRAIN-
MACHINE INTERFACES

By
Sung-Phil Kim

May 2005

Chair: Jose C. Principe
Major Department: Electrical and Computer Engineering

The role of decoding models in the design of brain-machine interfaces (BMIs) is to
approximate the mapping from the firing activity of the cortical neuronal ensemble to
associated behavior. The linear model, that in a statistical signal processing setting is
called the Wiener filter, has been the primary vehicle to estimate the mapping. One of the
purposes of this dissertation is to conduct an extensive comparative study of multi-input,
multi-output (MIMO) decoding models in two experimental BMI settings in which
monkeys perform dissimilar behavioral tasks. The issues in decoding model estimation
for BMIs include the large input dimensionality, the spatio-temporal neural firing
patterns, nonstationary, and the adequacy of the linearity assumption. These issues lead
us to concentrate our studies into four research directions; the topology of the models
(linear versus nonlinear), regularization both in space and time, preprocessing from
discrete events to continuous input variables, and ways to cope with the nonstationarity
present in the data. The comparison of the optimized linear and nonlinear MIMO models
with the Wiener filter based on generalization performance shows that the improvement, although statistically significant, is minor with respect to the baseline.

A second line of investigation deals with the analysis of motor cortex activity based on experimental BMI setups. Firstly, we propose an input based strategy called use non-negative matrix factorization (NMF) to uncover spatio-temporal patterns in neuronal ensembles correlated to behavior. The specific spatio-temporal patterns of neural activity can be determined from the NMF basis vectors using only the input data, and their temporal relationships with behavior can be extracted from the NMF encodings. Secondly, a real time neuronal subset selection method is developed to find a subset of neurons that is most relevant to kinematic trajectories at every sampling time instance. The method based on an on-line implementation of the LAR (Least Angle Regression) algorithm requires the availability of the desired response. The experimental analysis demonstrates the nonstationary characteristics of the relationship between the activity of neuronal ensemble and behavior.
CHAPTER 1
INTRODUCTION

The direct control of machines by thought has been rather close to fiction until recent developments in neuroscience which seek direct interfaces between brain and machines. This emerging field has been called brain-machine interfaces (BMIs). One of the clinical demands driving BMIs is restoring motor functions in ‘locked-in’ patients who suffer from paralysis caused by traumatic or degenerative lesions. In fact, there are more than 200,000 patients in the United States of America who live with partial or total permanent paralysis with 11,000 new cases each year [Nob99]. Eventually, BMIs may also impact the very paradigm of human computer interfaces.

Several research groups have demonstrated that subjects can control robotic arms or computer cursors on screen by using their brain activity [Car03, Cha99, Ken98, Mor99, Mus04, Ser02, She03, Tay02 and Wes00]. These demonstrations in rodents, primates, and human patients show promising ways to bypass spinal cord lesions. In these experiments, up to a hundred electrodes are chronically implanted in motor areas in the cortex to record the electrical activities of hundreds of neurons. The control signals for external devices are extracted by a series of signal processing modules including spike detection/sorting algorithms and decoding algorithms. This experimental BMI paradigm, which is illustrated in Fig. 1-1, relies on three basic elements. Long-term and stable recordings enable us to obtain a mass of neuronal activity through microelectrode arrays. A mathematical model extracts the information of motor parameters from neuronal activity recordings in real time. A prosthetic device such as a robotic arm
receives control signals from a mathematical model to coordinate the subject’s intended movement.

Figure 1-1. A system identification block diagram for BMIs.

This dissertation mainly focuses on building mathematical models in BMIs. These models utilize spike trains provided by spike sorting algorithms as inputs, and desired response of movement parameters such as hand position, velocity, or the gripping force which are synchronously recorded by optical sensors during motor performance of the subject. The design of these models can be viewed as a system identification problem [Hay96a]. Recent investigations in BMI modeling have demonstrated successful estimation of the transfer function from motor cortex neural firing patterns to hand movement trajectory of primates, with a relatively simple Wiener filter [Cha99; Mor99; Ser02 and Wes00]. If one thinks about the complexity of the motor system, starting from the intricate firing modulation of millions of cells in the cortex, passing through the added complexity of the spinal cord functionality up to the spatio-temporal firing of motor neurons that control each muscle fiber, it is rather surprising that a simple linear projection in the input space is able to capture the behavior of this complex system with correlation coefficients around 0.8 between the desired and actual trajectories. This leads
us to look from an optimal signal processing framework at the challenges and opportunities of this class of models for BMIs.

There are several challenges of the application based on an idea of the BMI setup. First, the spatio-temporal patterns in spike trains data are not fully known and thus cannot guide us in the proper way for designing the models. Second, this is a MIMO (multiple inputs multiple outputs) mapping problem, with a large dimensionality (i.e., for 100 neuronal inputs, the Wiener filter with 10 taps has 1,000 free parameters for each coordinate of outputs). Third, the statistics are not constant either in time or in space. Fourth, some neuronal firings are not related to the task and constitute therefore noise in the data. Fifth, there is no way of knowing if the true mapping is linear or nonlinear. In spite of all these difficult questions the linear model learns the trajectory with a mean correlation coefficient of 0.6 – 0.8; therefore it is instructive to undertake a systematic analysis of the issues to derive Wiener filters for BMIs.

**Review of BMI Signal Processing**

An approach to restore motor functions in paralyzed patients using direct interfaces between cortical motor areas and artificial actuators was first proposed by Schmidt [Sch80]. He proposed to connect from the electrical activities of cortical neuronal ensemble to an actuator to bypass spinal cord injuries.

Recently, Chapin and co-workers demonstrated that rats were trained to receive rewards of water drops by pressing a lever to control the rotation of a robotic arm [Cha99]. A linear model learned by least squares utilized the activities of 21-46 neurons in primary motor cortex (M1) as inputs to predict the motion of robot. Rats turned out to learn to control the robotic arm using only neuronal signals without moving arms.
Afterwards, other research groups joined the line of the study of experimental BMIs. Wessberg et al. [Wes00] in a joint research group including Duke University, SUNY, and MIT demonstrated a real time control of robotic arm using up to 100 neuronal activities. The Wiener filter or time delay neural network (TDNN) was designed to predict the 3D hand trajectories of food reaching movements using neuronal bin count data with a 100ms non-overlapping time windows embedded by a 10-tap delay line. Carmena et al. at Duke University also showed that with a relatively large number of cells (>100) monkeys could brain control a robot arm to perform two distinct different motor tasks including reaching and grasping [Car03]. In these experiments, monkeys could control a real robotic actuator through a closed-loop BMI. They also reported the change of the contributions of neuronal populations during learning.

Taylor et al. at Arizona State University presented a 3D cursor tracking BMI in their report [Tay02], where a monkey made arm movements in a 3D virtual environment to reach a randomly placed target. Using 18 cells from primary cortical area (M1), they investigated the effect of visual feedback on movements by comparing open-loop trajectories of hand controlled cursor movements and closed-loop trajectories of brain controlled cursor movements. A co-adaptive movement prediction algorithm based on a population vector method, which was developed to track changes in cell tuning properties during brain controlled movement, iteratively refines the estimate of cell tuning properties as a subject attempts to make a series of brain controlled movement. Other works on decoding algorithms in BMIs were reviewed in Schwartz et al. [Sch01]. In this review, parametric linear models including the population vector algorithm [Geo83] and the Wiener filter, and non-parametric methods including the maximum likelihood
estimate, the principal component analysis (PCA) [Isa00], and self-organizing feature
maps (SOFM) [Lin97] were introduced as motor-related information extraction
algorithms from neural activity for BMIs.

Serruya et al. in Donoghue laboratory at Brown University also demonstrated that
monkeys tracked a continuously moving visual object in a video monitor by moving a
manipulandum [Ser02]. The Wiener filter with 50ms bins embedded by 20 tap delay lines
was used to predict hand position from 7~30 M1 cell activities. They also showed that
time required to acquire targets using brain control was very similar to hand control. Wu
et al. in the same group proposed using a Kalman filter as a decoding model [Wu03] for
finding probabilistic relationship between motion and mean firing rates (for 140ms time
windows). They extended this Kalman filtering framework to build a mixture of linear
models using a switching Kalman filter model in which the hidden state variables were
estimated by the expectation-maximization (EM) algorithm [Wu04].

Andersen and co-workers in Caltech implanted microelectrode arrays in posterior
parietal cortex (PPC) which is assumed to be responsible for planning of movements
[And04, Mus04 and She03]. High-level signals related to a goal of movements were
decoded using the maximum likelihood estimate of cursor positions from ~ 40 neuronal
activities in PPC of monkeys. They demonstrated that neuronal activities in PPC could
provide information about movement plans; thus they can be used for various neural
prosthetic applications without moving limbs.

Kennedy et al. first demonstrated a human BMI by implanting a special electrode
in the human neocortex to extract signals to control a cursor on a computer monitor
[Ken98]. Using spike trains as input to a computer, severely disabled patients could learn to move a cursor.

Our group at the University of Florida in collaboration with Duke University has designed decoding models for 3D food reaching or 2D target reaching BMIs, including the Wiener filter and recursive multilayer perceptrons (RMLP) [San02a]. Based on the sensitivity analysis in the trained linear and nonlinear models, we improved the performance of models using only relevant neuronal activities [San03b]. Further development of switching multiple linear models combined by a nonlinear network was proposed by Kim et al. to increase prediction performance in food reaching [Kim03b]. Recently, Rao et al. demonstrated that echo state networks could be used as an alternative to nonlinear models such as RMLP or TDNN, with relatively uncomplicated training [Rao04].

Overall reviews for BMIs can be found in the following studies: [And04, Don02, Nic01, Nic03a and Sch04]. For overall reviews of brain-computer interfaces (BCIs), see Wolpaw et al. [Wol02] and Friechs et al. [Fri04].

**Approaches**

In this dissertation, we will address the following issues: First, we will apply the Wiener filter algorithm [Hay96a] to the BMI applications and show its performance in two types of training data: food reaching and tracking reaching experimental datasets. This algorithm will be the golden standard for the other adaptive methods developed. Then we will compare other adaptive algorithms that reach the same solution in the statistical sense for stationary data, but may handle the nonstationarity nature of the data better. We are referring to the least mean square algorithm (LMS) that will be implemented here in its normalized form (NLMS) [Hay96a].
The issue of the number of free parameters of the model will be handled by three different techniques. The first is the subspace Wiener filter, which first projects the input data using principal component analysis (PCA) [Hay96b], and then derives a Wiener filter to the desired response. Although PCA has been used as a major subspace projection method, it does not orient the projection to take advantage of the desired response structure. As an alternative, we propose a new idea of seeking subspace decomposition in the joint space through a hybrid subspace method, which combines the criterion of PCA and partial least squares (PLS) [Jon93 and Kim03a]. We also implement reduction in the number of degrees of freedom of the model by using a generalized feedforward filter based on the gamma tap delay line [Pri93], which has the ability to cover the same memory depth of the tap delay line with smaller filter order. The third method implemented uses on-line regularization based on the $L_p$-norm penalty [Has01], which decreases the values of unimportant weights through training. The problem of finding the optimal parameter for the penalty function will be addressed. The next issue covered in this paper relates to the adequacy of the linear modeling. We design a nonlinear mixture of switching, competitive linear models that implement a locally linear but globally nonlinear model [Kim03b]. This structure can be thought as a time delay neural network (TDNN) [Hay96b] that is trained in a different way to conquer the difficulty of training thousands of parameters with relatively small data sets.

An important contribution of BMIs to brain-related research fields is opening a new avenue for the experimental studies for the investigation of real time operation of neural systems in behaving animals [Nic03a]. For instance, using experimental BMIs, we may be able to explore the real-time nonstationary operations of neuronal ensemble in
association with behavior. Also, the cellular contributions in a large neuronal population to the motor parameter encoding can be analyzed through BMIs.

In a view of this respect, we investigate the properties of neuronal ensemble synchronized with behavior in BMIs using several approaches. First, we will seek a way to represent neuronal activity more efficiently in the context of BMI modeling. Through the multiresolution analysis [Mur04] for neural spike trains, we can construct a richer input space to possibly extract more encoded information, thus enhancing prediction models [Kim05c]. The issue of designing suitable models in this extended input feature space will be addressed. Second, we will demonstrate an approach to determine neuronal spatio-temporal patterns using nonnegative matrix factorization [Lee99]. This mathematical procedure, which has been introduced for image processing, can be utilized to extract spatio-temporal patterns of different neuronal populations without training of models [Kim05d]. Third, a real time neuronal subset selection algorithm is developed to find out which groups of neuronal activities exhibit relevance to a particular hand trajectory, and to investigate nonstationary characteristics of neuronal ensemble in time [Kim05b]. This selection scheme is developed based on linear filters used for BMIs.

**Outline**

The dissertation is organized as follows: The experimental BMIs paradigms and the descriptions of the recorded datasets are presented in chapter 2. We revisit the applications of the linear adaptive filters including the Wiener filter to BMIs in chapter 3. In chapter 4, several regularization methods are investigated to solve the problem of a large number of free parameters. In chapter 5, the technique of a nonlinear modeling using competitive multiple linear models is introduced and discussed. The experimental results and the comparisons of all the models for the two different behavioral tasks are
summarized in chapter 6. Further developments of BMI models based on the multiresolution analysis are demonstrated in chapter 7. Several analytical methods including NMF and on-line subset selection using experimental BMIs are introduced in chapter 8 and 9. Conclusions and future research directions are discussed in chapter 10.
CHAPTER 2
EXPERIMENTAL SETUPS FOR BRAIN-MACHINE INTERFACES

The datasets that are used for the prediction models were collected in experimental BMIs paradigm by Nicolelis lab at Duke University. In this paradigm, the electrical activity of cortical neuronal ensembles from awake, behaving primates were recorded and used by statistical models for controlling a robotic arm in which the arm movements of primates was reproduced. In this chapter, we describe the recording of the activity of neuronal ensembles and the experimental paradigm for behavioral tasks. The properties of the datasets are also presented.

Recording of Electrical Activity of Neuronal Ensembles

Multiple microwire arrays were chronically implanted in multiple cortical areas of one adult female owl monkey (*Aotus trivirgatus*) named as Belle, and two adult female Rhesus monkeys (*Macaca mulatta*) named as Ivy and Aurora. In an owl monkey, multiple low-density microelectrode arrays (MBlabs, Dennison, TX), each including 16-32 50-µm Teflon-coated stainless microwires, were implanted in the left dorsal premotor cortex (PMd), left primary motor cortes (M1), left posterior parietal cortex (PP), right PMd and M1, and right PP cortex [Wes00]. In the first Rhesus monkey (Aurora), multiple high-density microelectrode arrays developed at Duke University were implanted in the right PMd, right M1, right somatosensory (S1), right supplementary motor area (SMA), and the left M1 cortex. In the second Rhesus monkey (Ivy), multiple high-density microelectrode arrays were implanted in the right PP, M1, and SMA cortex [Car03 and Nic03b].
After surgical procedures, a multichannel acquisition processor (MAP, Plexon, Dallas, TX) cluster was used in the experiments to record the neuronal action potentials simultaneously. The spikes of single neuron from each microwire were discriminated based on time-amplitude discriminators and a principal component (PC) algorithm [Nic97 and Wes00]. Analog waveforms of the action potential and the firing time of each spike were stored. The firing times are binned within a 100ms nonoverlapping window, yielding a sequence of counts of the number of spikes in each bin. The distribution of the activity from the sorted neurons over cortex is presented in table 2-1 for each monkey. In this table, the indices of the sorted neuronal activity based on electrode arrays are used for identification purpose. These indices will be used through the remainder of dissertation. Note that in table 2-1, contra indicates the cortical areas in the opposite hemisphere to moving hand, ipsi does the areas in the same hemisphere.

Table 2-1. The distributions of the sorted neuronal activity for each monkey in motor cortical areas.

<table>
<thead>
<tr>
<th>Monkey</th>
<th>PP-contral</th>
<th>M1-contral</th>
<th>PMd-contral</th>
<th>S1-contral</th>
<th>SMA-contral</th>
<th>M1-ipsi</th>
<th>PMd / M1-ipsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belle</td>
<td>1-33</td>
<td>34-54</td>
<td>55-81</td>
<td>82-104</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(33)</td>
<td>(21)</td>
<td>(27)</td>
<td>(23)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ivy</td>
<td>1-49</td>
<td>50-139</td>
<td>140-192</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(49)</td>
<td>(90)</td>
<td>(53)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aurora</td>
<td>67-123</td>
<td>1-66</td>
<td>124-161</td>
<td>162-180</td>
<td>181-185</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(57)</td>
<td>(66)</td>
<td>(38)</td>
<td>(19)</td>
<td>(5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Behavioral Tasks**

During a recording period, each primate was trained to perform particular motor tasks. In the first experimental setup, an owl monkey (Belle) performed three-dimensional movements to reach for food randomly placed at one of four positions on a tray as depicted in Fig 2-1. In this task, the monkey placed its hand on a platform

---

1 The number of the sorted neuronal activity in the cortical area.
attached to the chair. When a barrier was open, the monkey reached and grabbed food.

The location and orientation of the wrist of the monkey were continuously recorded using a plastic strip with multiple fiber optic sensors (Shape Tape, Measureand, Inc., Fredricton, NB, Canada) [Wes00]. These signals were sampled at 200Hz.

![Image](image.png)

Figure 2-1. An experimental setup of 3D reaching task.

In the second experimental setup, the Rhesus monkeys (Aurora and Ivy) performed a two-dimensional target reaching task (Fig. 2-2). In this task, the monkey was cued to move the cursor on a computer screen by controlling a hand-held manipulandum in order to reach the target. The monkey was rewarded when the cursor intersected the target. The position of the manipulandum was continuously recorded at 1000Hz sampling rate.

**Properties of Data**

BMI models are designed to receive the binned spike counts as input signals and to predict hand position or velocity as desired signals. Before describing BMI models, it is informative to get the picture of the characteristics of input-output data. Therefore, we here present several characteristics of the data which are used for all BMI models in the remainder of this dissertation.
Figure 2-2. An experimental setup of 2D target reaching task. The monkey moves a cursor (yellow circle) to a randomly placed target (green circle), and rewarded if a cursor intersects the target.

**Neuronal Firing Patterns**

Firstly, the examples of the binned data are illustrated in Fig. 2-3 for six sample neurons collected from M1 cortex of Belle. We can notice that some neurons fire more frequently than others.

Figure 2-3. An example of the binned data.

Secondly, we examine the descriptive statistics of the binned data over entire neurons. The first statistic that we evaluate is the sparseness of the data measured by the ratio of the number of null bins (containing no spike) to the total number of bins. As a
result, the sparseness is 85.6% for Belle’s dataset, 65.2% for Ivy, and 60.5% for Aurora, respectively. Then, the average and the standard deviation of the bin count for each neuron are evaluated in three datasets as depicted in Fig. 2-4. It shows the variance of statistics over neuronal space.

Figure 2-4. The plots of the average (dot) and the standard deviation (bar) for each neuron of three monkeys, (a) Belle, (b) Ivy, and (c) Aurora are illustrated

In addition, the difference of firing rates during movement and rest for a 3D reaching task is evaluated. In order to quantify the difference, we estimate the mean firing rate during movement and rest separately. We collect 1300-second long contiguous data samples from Belle’s dataset, and manually select 81 subsets of movement from them. The remaining parts are referred to rest subsets. Then the mean firing rate of each subset
for movement and rest is estimated by averaging bin counts over entire neurons and time period of a given subset, respectively. Figure 2-5 shows the resulted estimates of mean firing rates for movement and rest. It shows that neurons tend to fire more frequently in average during movement. However, due to the uncertainty of the segmentation between movement and rest these average statistics are variable and subject to changing. It is also noteworthy that the mean firing rate tends to reduce with time.

Figure 2-5. The trajectories of the estimated mean firing rates for movement (solid line) and rest (dotted line) over sequence of subsets.

Finally, the nonstationary characteristics of input are investigated through observation of temporal change of the input autocorrelation matrix. The autocorrelation matrix of the multi-dimensional input data is estimated based on the assumption of ergodicity (see chapter 3 for details). In order to monitor the temporal change, the autocorrelation matrix is estimated for a sliding time window (4000-sample length) which slides by 1000 samples (100 second). For each estimated autocorrelation matrix, the condition number and the maximum eigenvalue are computed as approximations of the properties of the matrix. The experimental results of these quantities for three datasets are presented in Fig. 2-6. It is observed that there is temporal variance of the properties of the input autocorrelation matrix.
Hand Movements

The hand movements of primates are mainly parameterized by the trajectories of hand positions. We treat these trajectories as our desired signals to be predicted. Note that the hand positions which are sampled at 200Hz or 1000Hz are downsampled to 10Hz to be synchronized with the 100ms binned data. Before the investigation of the characteristics of desired signals, we first present the sample trajectories from two different tasks (food reaching of Belle and target reaching of Ivy) in Fig. 2-7. In the food reaching movement (Fig. 2-7a), the trajectory approximately spans a hyper-plane in which three specific parts of movement such as reach to food, food to mouth, and mouth to rest are placed. Figure 2-7a describes three reaching movements. In Fig. 2-7b, a 2D trajectory in the target reaching task over 4 second time duration is depicted. The trajectory starts from the dot in the middle of the figure to the arrow. It demonstrates the trajectory in this task spans the entire given 2D space and is more irregular than in 3D food reaching.
Figure 2-7. Sample trajectories of (a) 3D food reaching, and (b) 2D target reaching movements.

Now, we seek to observe the nonstationary characteristics of these trajectory signals. The continuous wavelet transform based on the basic wavelet function such as the Daubechies wavelet (db6 wavelet is used in this analysis) [Dau92] is performed to see the frequency change over time. 10000-sample trajectory data from both 3D food reaching and 2D target reaching are used for wavelet analysis. The absolute values of wavelet coefficients are plotted in Fig. 2-8. From this wavelet transform, we can clearly see the nonstationarity of the trajectory signals for both tasks.
Figure 2-8. The db6 continuous wavelet coefficients of trajectory signals of (a) 3D food reaching, and (b) 2D target reaching. Darker pixels in coefficients indicate larger values.
CHAPTER 3
LINEAR MODELING

In this chapter, we will present the design of adaptive linear filters for BMIs and the standard methods to estimate the parameters.

Linear Modeling for BMIs

Consider a set of spike counts from \(M\) neurons, and a hand position vector \(d \in \mathbb{R}^C\) (\(C\) is the output dimension, \(C = 2\) or \(3\)). The spike count of each neuron is embedded by an \(L\)-tap time-delay line. Then, the input vector for a linear model at a given time instance \(n\) is composed as:

\[
x(n) = [x_1(n), x_1(n-1) \ldots x_1(n-L+1), x_2(n) \ldots x_M(n-L+1)]^T, \quad x \in \mathbb{R}^{L \cdot M},
\]

where \(x_i(n-j)\) denotes the spike count of neuron \(i\) at a time instance \(n-j\). A linear model estimating hand position at time instance \(n\) from the embedded spike counts can be described as:

\[
y^c = \sum_{i=0}^{L-1} \sum_{j=1}^{M} x_i(n-j)w_{ij}^c + b^c
\]

(\(3-1\))

where \(y^c\) is the \(c\)-coordinate of the estimated hand position by the model, \(w_{ij}^c\) is a weight on the connection from \(x_i(n-j)\) to \(y^c\), and \(b^c\) is a bias for the \(c\)-coordinate. The bias can be removed from the model when we normalize \(x\) and \(d\) such that \(E[x] = 0, 0 \in \mathbb{R}^{L \cdot M}\), and \(E[d] = 0, 0 \in \mathbb{R}^{C}\), where \(E[\cdot]\) denotes the mean operator. Note that this model can be regarded as a combination of three separate linear models estimating each coordinate of hand position from identical input. In a matrix form, we can rewrite (1) as:

\[
y = W^T x
\]

(\(3-2\))
where \( y \) is a \( C \)-dimensional output vector, and \( W \) is a weight matrix of dimension \((L \cdot M + 1) \times C\). Each column of \( W \) consists of \([w_{10}^c, w_{11}^c, w_{12}^c, \ldots, w_{1L-1}^c, w_{20}^c, w_{21}^c, \ldots, w_{M0}^c, \ldots, w_{ML-1}^c]\)\(^T\).

Fig. 3-1 shows the topology of the linear model for the BMI application, which will be kept basically unchanged in the reminder of this dissertation. The most significant differences will be in the number of parameters and in the way the parameters \( w_{ji} \) of the model are computed from the data.

All the models are applied to estimate the 3D or 2D hand positions using \( L = 10 \) taps, \( M = 99 \) (Belle) neurons (after eliminating the ones that do not fire during training parts of recordings) for the food reaching task and \( M = 192 \) (Ivy) or 185 (Aurora) for the target reaching task. The length of the time delays \( (L) \) is determined based on the preliminary BMI study of the correlation between time lags and hand movements in Wessberg et al. [Wes00], where the neuronal firings up to 1 second before current hand
movement are significantly correlated with movement. The sizes of the training and the testing sets are 10,000 samples (~16.7 minutes) and 3,000 samples (~ 5 minutes) for all the models and three datasets, respectively. The size of the training set is empirically chosen by consideration of the compromise between nonstationarity and the quality of estimation: a longer training set can improve estimation of parameters, but increases a chance of entering more nonstationary characteristics of data in estimation. The weights are fixed after adaptation, and the outputs of the model are produced for novel testing samples. Performance of the model is evaluated based on these testing outputs with respect to generalization.

The following quantitative performance measures are used to evaluate the accuracy of the estimation:

1. Correlation coefficient (CC) quantifies the linear relationship between estimated and actual hand trajectories defined as

\[ CC \equiv \frac{C_{dy}}{s_ds_y} \]  

where, \( C_{dy} \) denotes the covariance between two variables \( d \) and \( y \), and \( s_d \) (or \( s_y \)) denotes the standard deviation of \( d \) (or \( y \)). In our evaluation, \( C_{dy} \) is the covariance between actual hand trajectory (\( d \)) and its estimation by model (\( y \)).

2. The signal to error ratio (SER) is the ratio of the powers of actual hand trajectory signals and the error of a model defined as

\[ SER \equiv \frac{\sum_{k=1}^{K}|d(k)|^2}{\sum_{k=1}^{K}|e(k)|^2} \]  

where \( d(k) \) and \( e(k) \) are the actual hand signal and the error at a time instance \( k \), and \( K \) is the size of the window in which SER is computed.

3. The cumulative error metric (CEM) estimates the cumulative distribution functions of the error radius defined as

\[ CEM(r) = \Pr(||e|| \leq r) \]  

So, CEM(\( r \)) is the estimated probability that the radius of the error vector is less than or equal to certain value \( r \).
We compute CC and SER for a short sliding time window in order to see if a given model predicts better for a particular part of trajectory. The size of the window is determined empirically. For the food reaching data, the size is set to 4 seconds which single reaching movement approximately takes. However, the duration of movement cannot be estimated for target reaching data since there is no apparent rest period between consecutive reaching movements. Therefore, the size of the window for the target reaching data is set long enough (1 minute) to make the computation of CC and SER reliable.

For comparison between different models, the averages of CC and SER from every window are computed respectively. These computations are conducted separately for each coordinate of hand position. Furthermore, we divide the evaluation results of food reaching into two modes: movement and rest. In each mode, the averages of CC and SER over three coordinates are used for evaluation instead of individual CC and SER in each coordinate. For target reaching, where the separation between movement and rest is not apparent, evaluation is executed separately for each coordinate.

The three performance measures introduced here complement one another; CC measures linear covariance between actual and estimated trajectories, thus providing the evaluation of tracking ability. But it lacks measuring the bias of estimation. This shortcoming is supplemented by SER that is based on error measurement. However, SER bears a problem such that it can be inclined to coordinate system which is calibrated artificially. For instance, with the similar error power, SER becomes relatively large when the magnitude of actual trajectory becomes large, thus increasing signal power. However, the magnitude of hand position does not possess any practical meaning. This
problem can be counterbalanced by CEM in which only the radius of error vector is concerned. It also provides a statistical tool for performance measure which is especially useful for statistical comparison of model on average. Hence, we can state that three measures jointly allow more comprehensive performance evaluation than using individual measures separately.

The Wiener Filter

The transfer function from the neural bin count to hand position can be estimated by linear adaptive filters, among which the Wiener filter plays a central role [Hay96]. The weight matrix in the Wiener filter for the case of MIMO system is estimated by the Wiener-Hopf solution as

$$W_{\text{Wiener}} = R^{-1}P.$$  (3-6)

$R$ is the correlation matrix of neural spike inputs with the dimension of $(L \cdot M) \times (L \cdot M)$,

$$R = \begin{bmatrix}
    r_{11} & r_{12} & \cdots & r_{1M} \\
    r_{21} & r_{22} & \cdots & r_{2M} \\
    \vdots & \vdots & \ddots & \vdots \\
    r_{M1} & r_{M2} & \cdots & r_{MM}
\end{bmatrix},$$  (3-7)

where $r_{ij}$ is the $L \times L$ cross-correlation matrix between neurons $i$ and $j$ ($i \neq j$), and $r_{ii}$ is the $L \times L$ autocorrelation matrix of neuron $i$. $P$ is the $(L \cdot M) \times C$ cross-correlation matrix between the neuronal bin count and hand position as

$$P = \begin{bmatrix}
    p_{11} & \cdots & p_{1C} \\
    p_{21} & \cdots & p_{2C} \\
    \vdots & \ddots & \vdots \\
    p_{M1} & \cdots & p_{MC}
\end{bmatrix},$$  (3-8)

where $p_{ic}$ is the cross-correlation vector between neuron $i$ and the $c$-coordinate of hand position. The estimated weights $W_{\text{Wiener}}$ are optimal based on the assumption that the
error is drawn from white Gaussian distribution and the data are stationary. The predictor $\mathbf{W}_{\text{Wiener}}^T \mathbf{x}$ minimizes the mean square error (MSE) cost function,

$$J = E[\|\mathbf{e}\|^2], \mathbf{e} = \mathbf{d} - \mathbf{y}.$$ \hspace{1cm} (3-9)

Each sub-block matrix $\mathbf{r}_{ij}$ can be decomposed as

$$\mathbf{r}_{ij} = \begin{bmatrix} r_{ij}(0) & r_{ij}(1) & \cdots & r_{ij}(L-1) \\ r_{ij}(-1) & r_{ij}(0) & \cdots & r_{ij}(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{ij}(1-L) & r(2-L) & \cdots & r(0) \end{bmatrix}, \hspace{1cm} (3-10)$$

where $r_{ij}(\tau)$ represents the correlation between neurons $i$ and $j$ with time lag $\tau$. These correlations, which are the second order moments of discrete-time random processes $x_i(m)$ and $x_j(k)$, are the functions of the time difference $(m-k)$ based on the assumption of wide sense stationarity ($m$ and $k$ denote discrete time instances for each process).

Assuming that the random process $x_i(k)$ is ergodic for all $i$, we can utilize the time average statistics to estimate correlation. In this case, the estimate of correlation between two neurons, $r_{ij}(m-k)$, can be obtained by

$$r_{ij}(m-k) = E[x_i(m)x_j(k)] \approx \frac{1}{N-1} \sum_{n=1}^{N} x_i(n-m)x_j(n-k), \ \forall i, j \in \{1, \ldots, M\}. \hspace{1cm} (3-11)$$

The cross-correlation vector $\mathbf{p}_{ic}$ can be decomposed and estimated in the same way.

$r_{ij}(\tau)$ is estimated using equation (3-11) from the neuronal bin count data with $x_i(n)$ and $x_j(n)$ being the bin count of neurons $i$ and $j$ respectively. From equation (3-11), it can be seen that $r_{ij}(\tau)$ is equal to $r_{ji}(-\tau)$. Since these two correlation estimates are positioned at the opposite side against the diagonal entries of $\mathbf{R}$, the equivalence between $r_{ij}(\tau)$ and $r_{ji}(-\tau)$ leads the symmetry of $\mathbf{R}$. The symmetric matrix $\mathbf{R}$, then, can be inverted effectively by using the Cholesky factorization. This factorization reduces the computational
complexity for the inverse of $\mathbf{R}$ from $O(N^3)$ to $O(N^2)$ where $N$ is the number of parameters.

Notice that $\mathbf{R}$ must be a nonsingular matrix to obtain the solution from (3-3). However, if the condition number of $\mathbf{R}$ is very large, causing $\mathbf{R}$ to be close to a singular matrix, then the $\mathbf{W}_{\text{Wiener}}$ may be inadequately determined. This usually happens when the number of samples is too small, or the input variables are linearly dependent to each other. In such a case, we can reduce the condition number by adding an identity matrix multiplied by some constant to $\mathbf{R}$ before inversion. This procedure is called ridge regression in statistics [Hoe70], and the solution obtained by this procedure turns out to minimize a cost function which linearly combines the one in (3-9) and a regularization term. The details will be discussed in chapter 4. In our estimation of the Wiener solution, however, we do not employ this regularization scheme.

Figure 3-2 and 3-3 display the Hinton diagrams of the weights of the Wiener filter obtained by (3-6) for food reaching and target reaching, respectively. Each column of $\mathbf{W}_{\text{Wiener}}$ (i.e., the weight vector of the Wiener filter for each coordinate) is rearranged in a matrix form to show spatio-temporal structure of weight vectors. In this matrix form, the neuron indices are aligned in the x-axis and the time lags are in the y-axis. Note that the first row of the matrix corresponds to the zero lag (for the instantaneous neuronal bin counts), followed by the successive rows corresponding to the increasing lags (up to nine). In the Hinton diagram, white pixels denote positive signs, while black ones do negative signs. Also, the size of pixels indicates the magnitude of a weight.

From the Hinton diagram, we can probe the contribution of individual neurons to the output of the Wiener filter. For this purpose, the weights represented in the Hinton
diagrams are yielded from the input in which each neuronal bin count time series \( x_j(n) \) is normalized to have unit variance. Then, the value of a weight can represent the sensitivity of the filter output to the corresponding input [San03a]. Also, we can see the sign of the correlation between a particular neuronal input and the output. For instance, the weights for neurons indexed by 5, 7, 21, 23, and 71 exhibit relatively large positive values for food reaching (see Fig. 3-2), indicating that those neuronal activities are positively correlated with the output. On the other hand, the weights for neurons 26, 45, 74, and 85 exhibit large negative values indicating the negative correlation between neuronal inputs and the output. There are also some neurons for which the weights have positive and negative values (e.g. 14 and 93). It is possible from these diagrams to examine the significant time lags for each neuron in terms of the contribution to the filter output. For instance, in the case of neuron 7 or 93, the recent bin counts seem to be more correlated with the current output. However, for neuron 23 or 74, the delayed bin counts seem to be more correlated with the current output. Similar observations can be made for target reaching in Fig. 3-3.

Figure 3-2. The Hinton diagram of the weights of the Wiener filter for food reaching.
Figure 3-3. The Hinton diagram of the weights of the Wiener filter for target reaching.

**Stochastic Gradient Learning**

The underlying assumption of the Wiener filter is that the statistics of the data are time-invariant. However, in the nonstationary environment where the statistics of the data vary in time, the Wiener filter only uses the average statistics to determine weights. The normalized least mean squares (NLMS) algorithm, a modified version of the least mean squares (LMS) algorithm, can train weights effectively for nonstationary inputs by varying the learning rate [Hay96]. It utilizes the stochastic estimation of the power of input signals to adjust the learning rate at each time instance. The weights at a given time instance \( n \) are updated by NLMS as

\[
\mathbf{w}_{NLMS}^c(n + 1) = \mathbf{w}_{NLMS}^c(n) + \frac{\eta}{\gamma + \|\mathbf{x}(n)\|^2} e^c(n)\mathbf{x}(n),
\]

where \( \eta \) satisfies \( 0 < \eta < 2 \), and \( \gamma \) is a small positive constant. \( e^c(n) \) is an error sample for the \( c \)-coordinate and \( \mathbf{x}(n) \) is an input vector. If we let \( \eta(n) \equiv \eta(\gamma + \|\mathbf{x}(n)\|^2) \), then the NLMS algorithm can be viewed as the LMS algorithm with a time-varying learning rate such that,

\[
\mathbf{w}_{NLMS}^c(n + 1) = \mathbf{w}_{NLMS}^c(n) + \eta(n)e^c(n)\mathbf{x}(n).
\]
Although the weights in NLMS converge to the same solution as the Wiener filter in the statistical sense for stationary data and a time-varying learning rate, the solution will be different for the nonstationary data.

The weights of the linear filter for BMIs are estimated by NLMS with the settings of $\eta = 0.01$ and $\gamma = 1$. In the empirical analysis of the resulted outputs of this filter, we observed that for food reaching the accuracy of the estimation is improved compared to the Wiener filter, especially during rest (see the details of results in chapter 6). It means that the weights found a better compromise between the two very different characteristics of movement and rest. This improvement has been achieved because of the update rule (3-12) where the weights in NLMS are updated with a relatively high learning rate during rest since total firing count increases during movement (see Fig. 2-5). Thus, for the class of motor behaviors in which movement periods are separated by rest, the NLMS algorithm captures more information about rest positions than the Wiener filter.

**Other Linear Modeling**

For comparison with other linear models being proposed for BMIs, a Kalman filter is designed and its prediction performance is evaluated for the same data used in this dissertation. The Kalman filter, which estimates the internal state for a linear dynamical system [Kal60] and produces a generative model for the data, has been proposed to learn the dynamical nature of the biological motor system in BMIs [Wu03, San02b]. In the Kalman filtering framework, the system state includes the hand position, velocity and acceleration, and the observation includes the neuronal bin count. Based on the assumption of the linear relationship (with additive Gaussian noises) between the state and the observation, as well as the states at current and previous time instances, the
Kalman filter recursively estimates the hand kinematics in real-time from cortical neurons. Although the system parameters representing the linear relationship are fixed after training, the Kalman filter can adjust its gain to track the time-varying nature of motor systems.

We briefly review the method of the Kalman filter used for BMIs. The linear dynamic equation for the state is given by

\[ z(n + 1) = Az(n) + \omega(n), \quad (3-14) \]

where \( z(n) \) is a state vector for the hand kinematics such that \( z(n) = [p_x(n) \ p_y(n) \ v_x(n) \ v_y(n) \ a_x(n) \ a_y(n)]^T \); \( p_c(n) \) denotes hand position for the \( c \)-coordinate, \( v(n) \) velocity, and \( a(n) \) acceleration, at a time instance \( n \). For food reaching, \( p_z(n), v_z(n), \) and \( a_z(n) \) are added to the state vector, respectively. \( \omega(n) \) is a process noise vector following the Gaussian distribution with a zero-mean vector and a covariance matrix \( \Omega \). The state-output mapping equation is given by

\[ x(n) = Hz(n) + \nu(n), \quad (3-15) \]

where \( x(n) \) is the instantaneous neuronal bin count vector (binned by a 100ms non-overlapping time window). Note that Wu et. al. designed the same Kalman filter with different window size (70ms) [Wu03]. \( \nu(n) \) is a measurement noise term following the Gaussian distribution with a zero-mean vector and a covariance matrix \( Q \). Given the training set, \( A \) and \( H \) are determined by the least squares (LS) which solves the following optimization problems,

\[ A = \arg\min_A \sum_{n=1}^{N-1} \|x(n+1) - Az(n)\|^2 \quad (3-16) \]

\[ H = \arg\min_H \sum_{n=1}^{N} \|x(n) - Hz(n)\|^2 \quad (3-17) \]
Given \( A \) and \( H \), the estimate of covariance matrix \( \Omega \) and \( Q \) can be obtained by

\[
\Omega = \frac{1}{N-1} \sum_{n=1}^{N-1} (z(n+1) - Az(n))(z(n+1) - Az(n))^T
\]

\[
(3-18)
\]

\[
Q = \frac{1}{N} \sum_{n=1}^{N} (x(n) - Hz(n))(x(n) - Hz(n))^T
\]

\[
(3-19)
\]

With the model \((A, H, \Omega, Q)\) obtained, the Kalman filter estimates the state of the hand kinematics from the novel neuronal bin count vectors (the test data) in real time. The state estimate \( z(n) \) and the Kalman gain matrix \( K(n) \) are updated at each time instance by the following recursion,

\[
P^{-}(n) = AP(n-1)A^T + \Omega
\]

\[
(3-20)
\]

\[
K(n) = P^{-}(n)H^T(HP^{-}(n)H^T + Q)^{-1}
\]

\[
(3-21)
\]

\[
z(n) = Az(n-1) + K(n)(x(n) - HAz(n-1))
\]

\[
(3-22)
\]

\[
P(n) = (I - K(n)H)P^{-}(n).
\]

\[
(3-23)
\]

Note that the error covariance matrix \( P \) and the state vector estimate \( z \) must be initialized before starting this recursion.
CHAPTER 4
REGULARIZED LINEAR MODELING

In chapter 3, we have demonstrated the design of linear filters which can be adapted for BMI applications. Despite the intrinsic sophistications in the BMI system, the simple linear filter (which merely combines the weighted bin count inputs) could estimate the primate’s hand position fairly well, especially showing the ability of tracking low-frequency trajectory. Based on this fact, we seek an opportunity to improve the performance of linear models by importing advanced learning techniques. Among those, a class of regularization methods is preferred since it yields smoother function approximation in order to improve the generalization performance for BMI models.

In this chapter, we propose to use three different regularization approaches. The first approach reduces the input space dimension using subspace projection and subsequently operates the linear filter in the subspace. The second approach reduces the filter order in each neuronal channel by employing the gamma delay line. The third approach places constraints on the model parameter space to reduce the effective number of parameters. We will discuss the methodology, implementation and analysis of these regularization approaches in this chapter.

**Dimension Reduction Using Subspace Projection**

One of the challenges in the design of decoding models for BMIs is that some neurons’ firings are not substantially modulated during task performance, and they only add noise to the multi-channel input data. In addition, some neurons’ firings are correlated with each other; thus it may be advantageous to blend these inputs to improve
model performance. Subspace projection, which can reduce the noise and blend correlated input signals together, may curtail unnecessary firing signals by a proper projection matrix. It also reduces the number of degrees of freedom in the multi-channel data, and consequently decreases the variance of the model. Here, we introduce a hybrid subspace projection method which is derived by combining the criteria of principal component analysis (PCA) and partial least squares (PLS). Then, we will design the subspace Wiener filter based on this hybrid subspace projection for BMIs.

### A Hybrid Subspace Projection

PCA, which preserves maximum variance in the data, has been widely adopted as a projection method [Hay96b]. The projection vector $\mathbf{w}_{PCA}$ is determined by maximizing the variance of the projection outputs as

$$
\mathbf{w}_{PCA} = \arg \max_{\mathbf{w}} J^{PCA}(\mathbf{w}) = E[\|\mathbf{x}^T \mathbf{w}\|^2] = \mathbf{w}^T \mathbf{R}_s \mathbf{w}
$$

where $\mathbf{R}_s$ is the input covariance matrix computed over the neuronal space only (it is an $M \times M$ matrix where $M$ is the number of neurons). $\mathbf{x}$ is an $M \times 1$ instantaneous neuronal bin count vector. It has been well known that $\mathbf{w}_{PCA}$ turns out to be the eigenvector of $\mathbf{R}_s$ corresponding to the largest eigenvalues. Then an $M \times S$ projection matrix which constructs an $S$-dimensional subspace consists of $S$ eigenvectors corresponding to the $S$ largest eigenvalues. However, PCA does not exploit information in the joint space of both input and desired response. This means that there may be directions with large variance that are not important to describe the correlation between input and desired response (e.g., some neuronal modulations related to the monkey’s anticipation of reward might be substantial, but less useful for the direct estimation of movement parameters), but will be preserved by the PCA decomposition.
One of the subspace projection methods to construct the subspace in the joint space is PLS, which seeks the projection maximizing the cross-correlation between the projection outputs and desired response [Jon93]. Given an input vector \( \mathbf{x} \) and a desired response \( d \), a projection vector of PLS, \( \mathbf{w}_{PLS} \) maximizes the following criterion,

\[
\mathbf{w}_{PLS} = \arg \max_{\mathbf{w}} \quad J^{PLS}(\mathbf{w}) = E[(\mathbf{x}^T \mathbf{w})d] = E[\mathbf{w}^T (\mathbf{x}d)] = \mathbf{w}^T \mathbf{p},
\]

where, \( \mathbf{p} \) is defined as an \( M \times 1 \) cross-correlation vector between \( \mathbf{x} \) and \( d \). The consecutive orthogonal PLS projection vectors are computed using the deflation method [Hay96b].

There have been efforts to find a better projection which can combine properties of PCA and PLS. The continuum regression (CR), introduced by Stone and Brooks [Sto90], attempted to blend the criteria of the ordinary least square (OLS), PCA and PLS. Recently, we have proposed a hybrid criterion function similar to the CR, together with a stochastic learning algorithm to estimate the projection matrix [Kim03a]. The learned projection can be either PCA, PLS, or combination of them. A hybrid criterion function combining PCA and PLS is given by

\[
J(\mathbf{w}, \lambda) = \frac{(\mathbf{w}^T \mathbf{p})^{2\lambda} (\mathbf{w}^T \mathbf{R}_s \mathbf{w})^{1-\lambda}}{\mathbf{w}^T \mathbf{w}}.
\]

where, \( \lambda \) is a balancing factor between PCA and PLS. This criterion covers the continuous range between PLS (\( \lambda = 1 \)) and PCA (\( \lambda = 0 \)).\(^1\) Since the log function is monotonically increasing, the criterion can be rewritten as,

\[
\log(\hat{J}(\mathbf{w}, \lambda)) = \lambda \log(\mathbf{w}^T \mathbf{p})^2 + (1 - \lambda) \log(\mathbf{w}^T \mathbf{R}_s \mathbf{w}) - \log(\mathbf{w}^T \mathbf{w}) \quad (4-4)
\]

\(^1\) The CR covers OLS, PLS and PCA. However, since we are only interested in the case when subspace projection is necessary, OLS can be omitted in our criterion.
We seek to maximize this criterion for $0 \leq \lambda \leq 1$. There are two learning algorithms derived in [Kim03a] to find $w$ (one is based on gradient descent, and the other is based on the fixed-point algorithm), but we opt to use the fixed-point learning algorithm here due to its fast convergence and independence of learning rate. The estimation of $w$ at the $k+1$\textsuperscript{th} iteration in the fixed-point algorithm is given by

$$w(k+1) = (1-T)w(k) + T \left[ \frac{\lambda p}{w(k)^T p} + \frac{(1-\lambda)R_s w(k)}{w(k)^T R_s w(k)} \right]$$

(4-5)

with a random initial vector $w(0)$. $T (0<T<1)$ is a balancing parameter to remove the oscillating behavior near convergence. The convergence rate is affected by $T$ that produces a tradeoff between the convergence speed and the accuracy. Note that the fastest convergence can be obtained with $T = 1$. The consecutive projection vectors are also learned by the deflation method to form in each column of a projection matrix $W$.

After projection onto the subspace by $W$, we embed the input signal at each channel with an $L$-tap delay line and design the Wiener filter to estimate the hand position. Figure 4-1 illustrates the overall diagram of the subspace Wiener filter.

Figure 4-1. The overall diagram of the subspace Wiener filter. $y(n)$ denotes the estimated hand position vector. There are $L$-1 delay operators ($z^{-1}$) for each subspace channel.
Design of a Decoding Model Using the Subspace Wiener Filter

The hold-out cross-validation method [Bis95] is utilized to determine both the optimal subspace dimension (S) and λ simultaneously. 10,000 consecutive data samples are divided into 9,000 training and 1,000 validation samples for both the food reaching and target reaching tasks, respectively. The MSE over validation samples is computed after training for each set of (Si, λj), where Si ∈ {20, 21, ..., 60} and λj ∈ {0, 0.1, ..., 1}. In Fig. 4-2, the contour map of the computed MSE is depicted. The minimum MSE is found at (S, λ) = (37, 0.9) for food reaching and (S, λ) = (44, 0.6) for target reaching, respectively. The validation MSE also tends to be smaller for larger λ in the lower subspace dimension while the MSE levels are rather flat in the higher subspace dimension. This indicates that PLS plays a more important role in building a better subspace Wiener filter for the lower subspace dimension.

![Figure 4-2. The contour map of the validation MSE for (a) food reaching, and (b) target reaching. The darker lines indicate lower MSE levels.](image)

To investigate the difference between the subspaces by PCA and PLS further, the first three projection vectors are estimated by setting λ = 0 or 1 in (4-5) as presented in Fig. 4-3. Note that PLS yields separate vectors corresponding to each hand position coordinate since it utilizes desired response, while PCA needs only one projection.
regardless of coordinates. In food reaching, the projection vectors of PCA have large weights on the neurons that fire frequently. For instance, the neurons indexed as 42, 57, and 93 are empirically discovered to have the largest firing counts. Since the neural firing data is sparse, PCA attempts to build a subspace with frequently firing neurons in order to preserve the variance. On the other hand, the weights in PLS projection have larger values on different neurons which do not fire very frequently such as the neurons indexed as 7 and 23. From the Hinton diagram described in previous chapter (see Fig. 3-2), these neurons were discovered to significantly contribute to the output of the Wiener filter designed for BMIs. Therefore, PLS is able to utilize the information from important neurons that do not even fire very frequently by exploiting the information in the joint space. For target reaching, we can also observe that more neurons are involved in the projection vectors in PLS than PCA. The neurons with larger weights in the PCA projection, again, are observed to fire more frequently. It is interesting to observe that for target reaching, the subspace dimension obtained from the cross-validation is of the same order as the number of neurons obtained in the neuron dropping analysis performed in Sanchez et al. [San03b]. In fact, the number of important neurons, for which the correlation coefficient between model outputs and desired hand trajectories is maximized, is 35, which is close to the subspace dimension of 44.

The empirical measurements of performance for the test data using the subspace Wiener filter with the above parameters demonstrate that the generalization performance of the subspace Wiener filter for both tasks reach slightly higher level than those of the Wiener filter or the linear filter trained by NLMS (see chapter 6). We expect, however, much higher improvements using the subspace projection methods for larger datasets
Figure 4-3. The first three projection vectors in PCA for (a) food reaching, and (c) target reaching, and PLS for (b) food reaching, and (d) target reaching, respectively.

(more than 200 neurons; Carmena, J.M., Lebedev, M.A., & Nicolelis, M.A.L., unpublished observations) and anticipate that these techniques will be important in the foreseeable future when the number of simultaneously recorded neurons surpasses 1,000.

Parsimonious Modeling in Time Using the Gamma Filter

The large number of parameters in decoding models is caused not only by the number of neurons but by the number of time delays required to capture the history of the neuron firings over time. Although we use a 10-tap delay line in this study, the size of the delay line can be variable depending upon the bin size (e.g., if we use a 50ms time bin,
then the number of time lags increases to 20). Hence, it is desirable to represent the
temporal patterns of neuronal data through more efficient way to reduce the number of
taps.

A linear filter described in previous chapters can be decomposed into multiple
finite impulse response (FIR) filters arranged to every neuron. An FIR filter has
advantages of trivial stability and easy adaptation. However, the length of the impulse
response and the filter order are equivalent in an FIR filter. Hence, when a problem
requires a deep memory and a small number of parameters, an infinite impulse response
(IIR) system is more likely appropriate. However, the stability issue in the adaptation and
the non-convex error surface of an IIR filter yield nontrivial challenges for practical use.
A generalized feedforward filter provides a signal processing framework to incorporate
both FIR and IIR characteristics into single system by employing a local feedback
structure [Pri93]. As shown in Fig. 4-4, an input signal is delayed at each tap by a delay
operator defined by specific transfer function $G(z)$. Note that when $G(z) = z^{-1}$, it becomes
an FIR filter. The transfer function of an overall system $H(z)$ is stable when $G(z)$ is stable
since

$$H(z) = \sum_{k=0}^{K} w_k (G(z))^k,$$  \hspace{1cm} (4-6)

where $K$ is the number of taps. It has been shown that a generalized feedforward filter can
provide a trivial stability and easy adaptation while decoupling the memory depth from
the filter order.
Figure 4-4. An overall diagram of a generalized feedforward filter [Pri93]. $x_0(n)$ is an instantaneous input, and $y(n)$ is a filter output.

The gamma filter is a special case of the generalized feedforward filter with $G(z) = \mu(z-(1-\mu))$ where $\mu$ is a feedback parameter. The impulse response of the transfer function from an input to the $k^{th}$ tap, denoted as $g_k(n)$, is given by

$$g_k(n) = Z^{-1}(G_k(z)) = Z^{-1}\left(\frac{\mu}{z-(1-\mu)}\right)^k = \left(\frac{n-1}{k-1}\right)\mu^k (1-\mu)^{n-k} u(n-k) \quad (4-7)$$

where, $Z^{-1}(\cdot)$ indicates the inverse z-transform and $u(n)$ the step function. When $\mu = 1$, the gamma filter becomes an FIR filter. The stability of the gamma filter in adaptation is guaranteed when $0 < \mu < 2$ due to a local feedback structure.

The memory depth $D$ with a feedback parameter $\mu$ in the $K^{th}$-order gamma filter is given by

$$D = \frac{K}{\mu} \quad \text{for} \quad \mu < 1, \text{ or } D = \frac{K}{2-\mu} \quad \text{for} \quad \mu > 1. \quad (4-8)$$

If we defined the resolution $R \equiv \mu$, the property of the gamma delay line can be described as

$$K = D \times R \quad \text{for} \quad \mu < 1, \text{ or } K = D \times (2 - R) \quad \text{for} \quad \mu > 1. \quad (4-9)$$

This property shows that the gamma filter decouples the memory depth from the filter order by adjusting a feedback parameter ($\mu$). In the case of $\mu = 1$ (i.e., the FIR filter), the
resolution is maximized whereas the memory depth is minimized for a given filter order. But this choice sometimes results in overfitting when a signal to be modeled requires more time delays than the number of descriptive parameters. Therefore, the gamma filter with the proper choice of a feedback parameter can avoid overfitting by the decoupled memory structure.

The tap weights can be updated using NLMS, and therefore the computational complexity is of the same order of FIR filters. A feedback parameter $\mu$ can also be adapted from the data. However, instead of adaptively learning $\mu$, we can search the best combination of $K$ and $\mu$ by using the cross-validation. In the same way as performed in previous section, the MSE in a validation set is computed for each set of $K_j$ and $\mu_i$, where $K_j \in \{2, 3, \ldots, 10\}$ (note that we ignore the case of $K_j=1$, which implements memoryless process) and $\mu_i \in \{0.1, 0.2, \ldots, 1.9\}$. The number of samples is 9000 and 1000 for training and validation, respectively. The contour of the validation MSE is shown in Fig. 4-5. The minimum MSE is achieved for $(K, \mu) = (4, 0.3)$ for food reaching and $(K, \mu) = (10, 1.2)$ for target reaching, respectively.

The memory depth estimated by this empirical method becomes $D \approx 13$ for the food reaching task and $D \approx 12.5$ for the target reaching task. The savings in the number of parameters are 60% (3120 → 1248) for the food reaching task. It appears that the temporal resolution of the filter $(R)$ for target reaching is larger than that for food reaching: $R = 0.3$ for food reaching and 0.8 for target reaching, respectively. It might indicate that relatively irregular target reaching movement requires finer temporal resolution. The generalization performance of the gamma filter with the optimized $K$ and
Figure 4-5. The contour maps of the validation MSE computed at each grid \( \{K_j, \mu_i\} \) for (a) food reaching, and (b) target reaching. The darker lines denote lower MSE levels.

\( \mu \) is evaluated through the novel test data. The empirical results show that the gamma filter exhibits slightly better performance than both the Wiener filter, and the FIR filter trained by NLMS (see chapter 6).

**Regularization by Parameter Constraints**

There have been numerous efforts about model selection to deal with the bias-variance dilemma [Gem92]. One of them is a pruning technique which seeks to diminish unnecessary parameters by imposing some constraints in the model parameter space (see [Ree93] for review). Among many pruning techniques, weight decay has been widely used due to its simplicity and fair performance [Kro92]. Weight decay is based on an error cost function to which an additional penalty term of parameters is added. This penalty restricts the \( L_2 \)-norm of a parameter vector, and is balanced with the MSE cost by a regularization parameter. Although weight decay has originated in a neural networks field, it shares the same cost function with a statistical method called ridge regression [Hoe70]. A difference is that ridge regression provides an analytical solution, where weight decay provides an iterative solution. Hence, understanding ridge regression may
give us a better appreciation of weight decay. One of the interesting features of ridge
regression is its link to subspace projection, especially PCA. This feature leads us to see
in which directions in the input space ridge regression (or weight decay) prunes more.
This property will be reviewed in more details shortly.

Ridge regression belongs to a class of shrinkage method in statistical learning. As
stated earlier, it employs the $L_2$-norm penalty. However, recent studies of statistical
learning have revealed that the $L_1$-norm penalty sometimes provides a better solution as a
shrinkage method in many applications than the $L_2$-norm penalty [Has01]. LASSO
(Linear Absolute Shrinkage and Selection Operator) has been a prominent algorithm
[Tib96] among the $L_1$-norm based shrinkage methods. However, its implementation is
computationally complex. LAR (Least Angle Regression) has been recently proposed by
Efron et. al. providing a framework to incorporate LASSO and forward stagewise
selection [Efr04]. With LAR, the computational complexity in the learning algorithm can
be significantly reduced.

It is notable that we have already applied this class of regularization for BMIs using
NLMS since NLMS can be viewed as the solution to the constrained optimization
problem [Hay96a]. In fact, the NLMS algorithm described in (3-12) is the solution to the
following problem:

Minimize $\|w(n+1) - w(n)\|^2$
subject to $d(n) - w(n+1)^T x(n) = 0$

for a given desired response $d(n)$ and an input vector $x(n)$ [Dou94]. It has also been
shown in [Slo93] that NLMS can be the solution to the following optimization problem:

$$\min_{w(n+1)} \|d(n) - w(n+1)^T x(n)\|^2 + \left(\frac{1}{\mu} - 1\right)\|x(n)\|^2 \|w(n+1) - w(n)\|^2$$  \hspace{1cm} (4-11)
where $\mu$ is the step size. In the NLMS algorithm, the weights are updated such that the change of weight vectors is minimized. The NLMS algorithm can be therefore viewed as the solution to the error minimization problem with the constraints on the difference between successive weight updates.

In this section, we will review statistical shrinkage methods and its relationship with subspace projection. Then, the application of ridge regression and weight decay for BMIs will be investigated. Finally, the properties of the LAR algorithm and its application to BMIs will be discussed.

**Review of Shrinkage Methods**

Here, we review the basic concepts in coefficient shrinkage methods. The link between subspace projection and shrinkage methods is then illustrated. Various shrinkage methods are finally illustrated both in a geometric view and in a Bayesian framework.

**Shrinkage methods**

Consider a constrained minimization problem for given an input vector $x$, and an desired output $d$ such that

$$
\hat{w} = \arg \min_w \ E[\|d - w^T x\|^2],
$$

subject to $\|w\|^2 \leq \rho$

(4-12)

where $w$ is a linear model parameter vector, and $\hat{w}$ is the optimal solution to it. This modeling technique is called ridge regression. When there are many correlated input variables in a linear model, the estimated weights can be poorly determined with high variance. For instance, the effect of a large positive weight on an input variable can be canceled by a large negative weight on another correlated input variable. If we restrict the size of weights as in (4-12), such a problem can be effectively prevented. The other
motivation of ridge regression is to make the input autocovariance matrix nonsingular even if it is not of full rank. Let $\mathbf{X}$ be an $N \times L$ input matrix in which each row represents an observation vector ($\mathbf{x}$ in equation 4-12), and $\mathbf{d}$ be an $N \times 1$ desired output vector. $N$ indicates the number of observations, and $L$ is the input dimension. We assume that each column of $\mathbf{X}$ is normalized to have zero-mean. Then, the optimal solution $\hat{\mathbf{w}}$ in (4-12) by ridge regression is

$$\hat{\mathbf{w}}_{RR} = (\mathbf{R} + \delta \mathbf{I})^{-1} \mathbf{P},$$

(4-13)

where $\mathbf{I}$ is an $L \times L$ identity matrix. $\mathbf{R}$ and $\mathbf{P}$ represent $\mathbf{X}^T \mathbf{X}$ and $\mathbf{X}^T \mathbf{d}$, respectively. Notice that the matrix $\mathbf{R} + \delta \mathbf{I}$ is invertible even if $\mathbf{R}$ is a singular matrix.

We can obtain some insights in the properties of ridge regression by the singular value decomposition (SVD) of $\mathbf{X}$. The SVD of $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{U} \mathbf{A} \mathbf{A}^T,$$

(4-14)

where $\mathbf{U}$ and $\mathbf{V}$ are $N \times L$ and $L \times L$ unitary orthogonal matrices, and $\mathbf{A}$ is an $L \times L$ diagonal matrix with diagonal entries $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L \geq 0$ called the singular values. Then, the prediction outputs yielded by ridge regression can be written using the SVD as

$$\mathbf{X} \hat{\mathbf{w}}_{RR} = \mathbf{X}(\mathbf{X}^T \mathbf{X} + \delta \mathbf{I})^{-1} \mathbf{X}^T \mathbf{d} = \mathbf{U} \mathbf{A} (\mathbf{A}^2 + \delta \mathbf{I})^{-1} \mathbf{A}^T \mathbf{d} = \sum_{i=1}^{L} \mathbf{u}_i \frac{\lambda_i^2}{\lambda_i^2 + \delta} \mathbf{u}_i^T \mathbf{d},$$

(4-15)

where $\mathbf{u}_i$ is the $i^{th}$ column of $\mathbf{U}$. From (4-15), we can see that ridge regression finds the coordinates of $\mathbf{d}$ with respect to each orthonormal basis $\mathbf{u}_i$. Then it shrinks coordinates by $\frac{\lambda_i^2}{\lambda_i^2 + \delta} (\delta > 0)$. Therefore the coordinate with a smaller $\lambda$ will be shrunk more. It is easy to show that the singular values $\{\lambda_i\}$ indicate the variance of the principal components of
X [Has01]. Hence, the smaller singular value corresponds to the direction of smaller variance which is shrunk more by ridge regression.

Now we consider LASSO as an $L_1$-norm based shrinkage method. The fundamental difference between ridge regression and LASSO is the penalty in the cost function as,

$$
\mathbf{w}_{lasso} = \arg \min_{\mathbf{w}} \mathbb{E}[\| \mathbf{d} - \mathbf{w}^T \mathbf{x} \|^2] \quad \text{subject to} \quad \sum_{i=1}^{L} |w_i| \leq \rho
$$

(4-16)

where $w_i$ is the $i^{th}$ element of weight vector $\mathbf{w}$. The solution to this minimization problem is no longer linear in $\mathbf{d}$, and a quadratic programming algorithm is usually used to compute the solution. The $L_1$-norm penalty in (4-16) can make some weights be exactly zero; thus LASSO is able to select a subset of inputs.

**The relationship between subspace projection and ridge regression**

We have seen that ridge regression shrinks all directions of principal components of $\mathbf{X}$, with different rates of shrinkage depending on the variance of each direction. Subspace projection with PCA, on the other hand, selects $S$ (subspace dimension) high-variance directions while ignoring the rest. PLS tends to shrink low-variance directions, while also reduce high-variance directions depending on the environments [Fra93]. It is obvious from these facts that a hybrid subspace projection utilized for BMIs would behave in a similar fashion as PCA and PLS. Hence, we can see that ridge regression and the hybrid subspace projection manipulate solutions in similar manner: they tend to shrink low-variance of principal directions more. The difference is that ridge regression shrinks smoothly while subspace projection shrinks in discrete steps.

**Comparison of shrinkage methods**

Generalization of ridge regression and LASSO creates a criterion as,
\[
\hat{w} = \arg\min_w \left[ E[\|d - w^T x\|^2] + \delta \sum_{i=1}^L |w_i|^p \right]
\]  
(4-17)

The penalty is $L_p$-norm for $p \geq 0$. In Fig. 4-6, the contours of $\sum |w_i|^p$ are illustrated in the two-dimensional weight space. Note the difference of contour shapes between ridge regression and LASSO. Since the contour for LASSO has corners, it is possible that the performance surface hits the corner, causing one weight to be zero. If the dimension of the parameter space increases, the contour shape for LASSO becomes a rhomboid, and has more corners, flat edges and faces. Then, there are more chances to generate zero coefficients. This geometric description illustrates why LASSO provides a sparser solution, including zero coefficients, than ridge regression.

![Figure 4-6. Contours of the $L_p$-norm of weight vector for various values of $p$ in the 2D weight space.](image)

(a) $p = 4$  
(b) $p = 2$  
(c) $p = 1$  
(d) $p = 0.5$

Now let us look at the criterion (4-17) in a Bayesian framework. The penalty term can be considered to represent a log-prior probability density function for $w_i$, with zero-mean and variance of $1/\delta$ [Nea96]. The prior distribution of $w_i$ is different depending on $p$. The $L_0$-norm simply counts the number of nonzero parameters. This corresponds to subset selection of input variables [Fur74]. The $L_1$-norm penalty has a Laplacian prior. The $L_2$-norm penalty has a Gaussian prior. Hence we can consider ridge regression, LASSO, and subset selection as Bayesian estimate of solution to (4-17) with different priors for the weight.
Regularization Based on the $L_2$-Norm Penalty

So far, the basic properties of shrinkage methods including ridge regression and LASSO have been investigated. The applications of these methods to BMI models will be discussed in the remainder of this chapter.

We have seen that an additional identity matrix scaled by white noise power to the input autocovariance matrix avoids singularity and help shrink input variables in the direction of the eigenvectors corresponding to smaller eigenvalues. However, it is an open problem to determine the noise power, or a so-called regularization parameter ($\delta$ in equation 4-13). Even if we want to determine the regularization parameter empirically, we need to follow a systematic procedure. One of the most popular procedures is the cross-validation, but it expenses a separate validation set and is not adequate in real-time procedure. For the real-time implementation of BMI models, therefore, we need a different procedure without generating an explicit validation set. One feasible approach is to maintain the balance between the noise power represented by the regularization parameter and the input signal power estimated by eigenvalues. In this approach, the input signal to noise power ratio (SNR) is estimated by

$$SNR \approx \frac{tr[R]}{\delta}$$  \hspace{1cm} (4-18)

where $tr[R]$ denotes the trace of the input covariance matrix $R$. From this estimation, we can approximate $\delta$ as,

$$\delta \approx \frac{tr[R]}{SNR}$$  \hspace{1cm} (4-19)

for a desirable SNR. For instance, if we want to ensure that the input SNR is kept greater than 30dB with $tr[R]$ computed as 0.1, then the regularization parameter is determined to
be $10^4$. This estimation procedure for the regularization parameter will be particularly useful in BMI implementation when we seek the analytical estimate of the parameters of a linear filter in real-time with a large number of neurons for which the inversion of the input autocorrelation matrix is not guaranteed.

Weight decay can be viewed as a simple on-line method to minimize criterion function in (4-17) using the stochastic gradient, updating the weights by

$$w(n + 1) = w(n) + \eta_w \hat{V} \zeta(n) - \delta w(n)$$  \hspace{1cm} (4-20)

where, $\hat{V} \zeta(n) = \partial E \left[ e(n) \right]^2 \partial w(n)$, and $\eta_w$ is a learning rate for the weight vector.

Instead of determining $\delta$ by the input SNR, we opt to use an adaptive procedure to estimate the optimal value from data. Larsen et al. [Lar96] proposed that $\delta$ can be optimized by minimizing the generalization error with respect to $\delta$. Following this procedure, we utilize the $K$-fold cross-validation [Gei75], which divides the data into $K$ randomly chosen disjoint sets, to estimate the average generalization error empirically as

$$\hat{\xi} = \frac{1}{K} \sum_{k=1}^{K} e_k,$$  \hspace{1cm} (4-21)

where $e_k$ is the validation MSE for the $k^{th}$ set. Then, the optimal regularization parameter is learned by using gradient descent as,

$$\delta(k + 1) = \delta(k) - \eta \frac{\partial \hat{\xi}(k)}{\partial \delta},$$  \hspace{1cm} (4-22)

where $\hat{\xi}(k)$ is the estimate of $\hat{\xi}$ at the $k^{th}$ iteration, and $\eta$ is a learning rate for the regularization parameter. The detail procedure of estimation of $\partial \hat{\xi}(n)/\partial \delta$ using weight decay is given in Larsen et al. [Lar96].
In the experiment, we set $K = 10$, $\eta = 10^{-6}$ and update $\delta$ until the difference 
\[ |\hat{x}(n+1) - \hat{x}(n)| \] 
becomes less than $10^{-3}$. The number of training samples is 9,000 and the number of validation samples is 1,000. The term $\hat{V}_\zeta(n)$ in (4-20) is estimated by NLMS. During training, $\delta$ converges to $1.36 \times 10^{-5}$ for food reaching and $1.02 \times 10^{-5}$ for target reaching, respectively, as depicted in Fig. 4-7. Then, we train the filter with $\delta$ fixed using the entire training samples (10,000) to obtain the regularized model. The histogram of the weight magnitude computed over all the coordinates of hand position is depicted in Fig. 4-8 to demonstrate the effect of weight decay. Note that the number of weights that have smaller magnitudes increases with weight decay. For instance, the number of weights that are close to zero is approximately 345 for weight decay versus 75 for NLMS in Fig. 4-8a, and 460 for weight decay versus 150 for NLMS in Fig. 4-8b. It shows that more weights are pruned by weight decay, thus the effective degree of freedom of the model reduces. The reduced degree of freedom can help generalization as examined by measuring performance in the test data. Empirical performance measures in the test dataset show that regularization using weight decay improves the generalization performance over the

![Figure 4-7](image.png)

Figure 4-7. Convergence of the regularization parameter $\delta(n)$ over iterations; (a) food reaching, and (b) target reaching.
Figure 4-8. The histogram of the magnitudes of weights over all the coordinates of hand position, trained by weight decay (solid line) and NMLS (dotted line); (a) food reaching, and (b) target reaching.

linear model trained only by NLMS (see chapter 6).

**Regularization Based on the $L_1$-norm Penalty**

The least angle regression (LAR) algorithm has been recently developed to accelerate computation and improve performance of forward model selection methods. It has been shown in Efron et al. that simple modifications to LAR can implement the LASSO and the forward stagewise linear regression [Efr04]. Essentially, the LAR algorithm requires the same order of computational complexity as the ordinary least squares (OLS).

The selection property of LAR, which leads to zeroing coefficients, is preferable for identification of sparse systems when compared to regularization methods with the $L_2$-norm penalty. Also, the analysis of the selection process often provides better insights into the unknown system than the $L_2$-norm based shrinkage methods.

The LAR procedure starts with an all zero coefficients initial condition. The input variable having the most correlation with desired response is selected. We proceed in the direction of the selected input with a step size which is determined such that some other
input variable becomes to have as much correlation with the current residual as the first input. Then, we move in the equiangular direction between these two inputs until the third input has the same correlation. This procedure is repeated until either all input variables join the selection, or the sum of coefficients meets a preset threshold (constraint). Note that the maximum correlation between inputs and the residual decreases over successive selection step in order to de-correlate the residual with inputs.

Table 4-1 summarizes the details of the LAR procedure [Efr04].

An illustration in Figure 4-9 (cited from Efron et al. [Efr04]) would help understand how the LAR algorithm proceeds. In this figure, we start to move on the first selected input variable $x_1$ until the next variable ($x_2$ in this case) has the same correlation.

Table 4-1. Procedure of the LAR algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Compute the current correlation $c = X^T (Y - \hat{Y})$</td>
</tr>
<tr>
<td>(b)</td>
<td>Find $C_{\text{max}} = \max_j {c_j}$, and a set $A = {j:</td>
</tr>
<tr>
<td>(c)</td>
<td>Let $X_a = {\ldots, \text{sign}(c_j)x_j, \ldots}$ for $j \in A$.</td>
</tr>
<tr>
<td>(d)</td>
<td>Let $\Phi = X_a^T X_a$, and $\alpha = (1_a^T \Phi^{-1} 1_a)^{-1}$, where $1_a$ is a vector of one’s with a length equal to size of $A$.</td>
</tr>
<tr>
<td>(e)</td>
<td>Compute the equiangular vector $\mu = X_a(\alpha \Phi^{-1} 1_a)$ that has the unit length. Note that $X_a \mu = \alpha 1_a$ (angles between all inputs in $A$ and $\mu$ are equal).</td>
</tr>
<tr>
<td>(f)</td>
<td>Compute the step size, $\gamma = \min_{j \in A}^+ \left{ \frac{C_{\text{max}} - c_j}{\alpha - \theta_j}, \frac{C_{\text{max}} + c_j}{\alpha + \theta_j} \right}$ where $\min^+$ indicates considering only positive minimum values over possible $j$.</td>
</tr>
<tr>
<td>(g)</td>
<td>Compute $\theta_j$ which is defined as the inner product between all inputs and $\mu$ such as, $\theta_j = X^T \mu$</td>
</tr>
<tr>
<td>(h)</td>
<td>Update $\hat{Y}_+ = \hat{Y} + \gamma \mu$.</td>
</tr>
</tbody>
</table>

Repeat a-h until all inputs join the active set $A$, or $\sum_j |\beta_j|$ exceeds the given threshold.
with the residual generated by \( x_1 \). \( \mu_1 \) is the unit vector in this direction as computed in table 4-1(e). The amount of movement along \( \mu_1 \), denoted as \( \gamma_1 \), is computed by equation in table 4-1 (f). \( y_1 \) denotes the OLS estimate of desired response \( y \) with input \( x_1 \). Note that the estimate by LAR (\( \hat{y}_1 \)) moves toward \( y_1 \), but does not reach it. The next direction \( \mu_2 \) intersects the angle between \( x_1 \) and \( x_2 \) (equiangular vector in the two-dimensional space of \( x_1 \) and \( x_2 \)) such that the angle between \( x_1 \) and the updated residual \( (r_1 = y_2 - \gamma_1 \mu_1) \) is same as the one between \( x_2 \) and \( r_1 \). Since every input is standardized such that the correlation, which is measured by the inner product of \( x_1 \) and \( x_2 \), can be estimated by the angel between \( x_1 \) and \( x_2 \), these two variables have the same absolute correlation with \( r_1 \) following the equation in Table 4-1(a). The coefficient \( \gamma_2 \) is computed again following Table 4-1(f) such that \( x_3 \) has the same absolute correlation with the next residual \( r_2 = y_3 - (\gamma_1 \mu_1 + \gamma_2 \mu_2) \) as \( x_1 \) and \( x_2 \). So, the next direction \( \mu_3 \) intersects the angle between \( x_1 \), \( x_2 \), and \( x_3 \). This procedure is repeated until the \( L_1 \)-norm of coefficients reaches a given threshold.

LAR can be easily modified to implement LASSO; when some coefficients cross zero at a given step, those are forced to be zero. And the corresponding inputs are

Figure 4-9. An illustration of the LAR procedure.
removed from the selected joint set. The LAR procedure can be continued with the remaining inputs since they still have the same absolute correlation with the current residual.

There are two major considerations in the implementation of LAR. First, LAR assumes the linearly independence between input variables. Second, the determination of threshold for the $L_1$-norm of coefficients is an open problem and dependent upon data. The performance of the linear model learned by LAR can be greatly influenced by a choice of this threshold.

If we attempt to apply LAR to the linear model in BMIs, difficulties lie in the fact that the embedded inputs are likely to be correlated with each other (although they might be linearly independent), so that LAR might not be able to operate optimally. Also, finding an optimal threshold will be a nontrivial task.\(^2\)

Despite these difficulties, we test the performance of the linear model learned by LAR with the food reaching and the target reaching datasets. The threshold is determined by the hold-out cross-validation. The performance measures computed in test data show that LAR performs in the similar level as weight decay. It may indicate that the difficulties in the implementation of LAR could prevent it from improving generalization further compared to weight decay. We will skip the presentation of the numerical performance results of the linear models with LAR since they are very similar to those with weight decay.

\(^2\) We can utilize the cross-validation as in the case of the gamma filter or subspace projection. However, the range of the search for threshold will become much broader.
CHAPTER 5
NONLINEAR MIXTURE OF MULTIPLE LINEAR MODELS

In the design of decoding models for BMIs, there have been a number of approaches including linear and nonlinear models; e.g. the Wiener filter, the Kalman filter, time delay neural networks (TDNN), recursive multilayer perceptrons (RMLP), and so on. These modeling frameworks have successfully predicted target hand trajectories only using neuronal activity signals.

However, an important consideration in designing BMIs is the feasibility of the approach taken. The target applications necessitate real-time implementations with minimal computational and hardware requirements. On one hand, linear models are usually the best in terms of their computational requirements. On the other hand, a simple linear model is often insufficient to accurately capture the complex input-output relationships between neural activity and hand position. Recently, a performance comparison has been conducted between linear and nonlinear modeling approaches, and the latter was found to be favorable [San02a].

In this chapter, we aim to demonstrate that the target mapping between the neural activity and the hand trajectories can be discovered using a divide-and-conquer approach. In this approach, we combine the simplicity of training linear models with the performance boost that can be achieved by nonlinear methods. Specifically, a two-stage structure is used where the first stage consists of a bank of competitively trained linear filters and the second stage consists of a single-hidden layer multilayer perceptrons (MLP) (see Fig. 5-1). Model comparison in the next chapter will demonstrate the
outstanding performance of this approach among various models for the food reaching BMI data.

**Nonlinear Mixture of Linear Models Approach**

In this section, we describe the modeling approach using nonlinear mixture of competitive linear models (NMCLM). A brief description of TDNN will also be provided for comparison purpose.

**Nonlinear Mixture of Competitive Linear Models**

The overall architecture of NMCLM is identical to a single hidden layer TDNN as shown in Fig. 5-1. However, the training procedure undertaken here is significantly different. This modeling method uses the *divide-and-conquer* approach. Our reasoning is that a complex nonlinear modeling task can be elucidated by dividing it into simpler linear modeling tasks and combining them properly [Far87]. Previously, this approach was successfully applied to nonstationary signal segmentation, assuming that a nonstationary signal is a combination of piecewise stationary signals [Fan96].

Hypothesizing that the neural activity will demonstrate varying characteristics for different localities in the space of the hand trajectories, we expect the multiple model approach, in which each linear model specializes in a local region, to provide a better overall input-output mapping. However, the problem is different here since the goal is not to segment a signal but to segment the joint input/desired signal space.

The topology allows a two-stage training procedure that can be performed *sequentially* in off-line training; first, competitive learning for the local linear models and then error backpropagation learning for the MLP. It is important to note that in this scheme, both the linear models and the MLP are trained to approximate the same desired response, which is the hand trajectory of a primate.
The training of the multiple linear models is accomplished by competitively (hard or soft competition) updating their weights in accordance with previous approaches using the NLMS algorithm. The winning model is determined by comparing the (leaky) integrated squared errors of all competing models and selecting the model that exhibits the least integrated error for the corresponding input [Fan96]. The leaky integrated squared error for the $i^{th}$ model is given by

$$\varepsilon_i(n) = (1 - \mu)\varepsilon_i(n-1) + \mu e_i^2(n), i = 1, \cdots, M$$

(5-1)

where, $M$ is the number of models and $\mu$ is the time constant of the leaky integrator.

Then, the $j^{th}$ model wins competition if $\varepsilon_j(n) < \varepsilon_i(n)$ for all $i \neq j$. If hard competition is employed, only the weight vector of the winning model is updated. Specifically, if the $j^{th}$ model wins competition, the update rule for the weight vector $w_j(n)$ of that model is given by
\[ w_j(n+1) = w_j(n) + \eta \frac{e_j(n)x(n)}{\gamma + \|x(n)\|^2} \]  

(5-2)

where, \( e_j(n) \) is the instantaneous error, and \( x(n) \) is the current input vector. \( \eta \) represents a learning rate and \( \gamma \) is a small positive constant used for normalization. If soft competition is used, a Gaussian weighting function centered at the winning model is applied to all competing models. Every model is then updated proportional to the weight assigned to that model by this Gaussian weighting function such that

\[ w_i(n+1) = w_i(n) + \eta \frac{\Lambda_{i,j}(n)e_j(n)x(n)}{\gamma + \|x(n)\|^2}, i = 1, \cdots, M \]  

(5-3)

where, \( w_i \) is the weight vector of the \( i^{th} \) model. Assuming the \( j^{th} \) model wins competition, \( \Lambda_{i,j}(n) \) is the weighting function defined by

\[ \Lambda_{i,j}(n) = \exp\left(-\frac{d_{ij}^2}{2\sigma^2(n)}\right) \]  

(5-4)

where, \( d_{ij} \) is the Euclidean distance between indices \( i \) and \( j \), which is equal to \( |j-i| \), \( \eta(n) \) is the annealed learning rate, and \( \sigma^2(n) \) is the Gaussian kernel width decreasing exponentially as \( n \) increases. The learning rate also exponentially decreases with \( n \).

Soft competition preserves the topology of the input space, updating the models neighboring the winner; thus it is expected to result in smoother transitions between models specializing in topologically neighboring regions (of the state space). However, the empirical comparison using BMIs data between hard and soft competition update rules shows no significant difference in terms of model performance (possibly due to the nature of the data set). Therefore, we prefer to utilize hard competition rule for its simplicity.
With the competitive training procedure, each model can specialize in local regions in the joint space. Figure 5-2 demonstrates the specialization of 10 trained models by plotting their outputs (black dots) with the common input data (40 seconds long) in the 3D hand trajectory space. Each model’s outputs are simultaneously plotted on top of the actual hand trajectory (red lines) synchronized with the common input. The figure shows that the input-output mappings learned by each model display some degree of localization, although overlaps are still present. These overlaps may be consistent with a neuronal multiplexing effect as depicted in Carmena et al. [Car03], which suggests that the same neurons modulate for more than one motor parameter (the x- and y-coordinates of hand position, velocity and gripping force).

The competitive local linear models, however, require additional information for switching when applied to BMIs, since the desired signal that is necessary to select a winning model is not available after training in practice. A gate function as in the mixture of experts [Jac91] utilizing input signals needs to be trained to select a local model. Here, we opt for a MLP that directly combines the predictions of all models. Therefore, the overall architecture can be conceived as a nonlinear mixture of competitive linear models.
(NMCLM) [Kim03b]. This procedure facilitates training of each model compared to the TDNN, since only one linear model is trained at a time in the first stage, while only a relatively small number of weights are trained by error backpropagation [Hay96b] in the second stage.

**Time Delay Neural Networks**

In the TDNN, the mapping between neural activity and hand trajectories is estimated by nonlinearly combining bin counts (and their past values) from each neuron. The tap delay lines in the input layer preset the memory to account for temporal dependencies in neural activity. This architecture has a single hidden layer with sigmoid nonlinearities, and the output layer with linear processing elements (PEs). The output of the TDNN is given by
\[
y(n) = W_2f(W_1^T x(n) + b_1) + b_2,
\]
where the weight matrices and bias vectors \(W_1, W_2, b_1,\) and \(b_2\) are trained by the error backpropagation algorithm.

**BMIs Design Using NMCLM**

NMCLM is trained with the same sets of data used for the Wiener filter in chapter 3. The topology consists of 10 competitive linear models for each coordinate and a single hidden layer MLP with \(M\) inputs \((M = 10 \cdot C, C\) is the output dimension: 2 or 3), 30 hidden PEs with the hyper-tangent (tanh) functions, and \(C\) linear output PEs to predict each hand position coordinate. Each linear model has the same topology as the one used in chapter 3. The number of multiple models and the number of hidden PEs were chosen empirically (although it was not optimized). The hard competition learning rule is utilized along with NLMS for the training of linear models and the conjugate gradient algorithm is used to train the MLP. The training of the MLP is repeated with 100 random initial conditions and the network with the least MSE is selected. The time constant of the leaky integrator \((\alpha)\) is determined by the hold-out cross-validation method. The data is divided...
into 9000-sample training set and 1000-sample validation set. The resulting values of $\alpha$ are 0.3 for the food reaching task and 0.6 for the target reaching task.

TDNN is trained with the same input and desired response as in NMCLM. The 30 PEs in the hidden layer use tanh nonlinearities. All the weights and biases are trained by the error backpropogation algorithm.

Even with the simpler training approach, there are over 30,000 parameters in NMCLM to be trained. Each linear model with around 3,000 parameters is trained with a fraction of the total number of samples (only the ones pertaining to its local area of the space), which is considered too high for the restricted number of training samples. With linear models built from gamma filters, we can reduce significantly the number of parameters in the first layer of NMCLM, while preserving the same level of computational complexity in training.

As will be shown in the next chapter, NMCLM results in superior generalization performance compared to other linear models and the TDNN for food reaching. Substitution of the gamma filters for the FIR filters also improves the performance further. Due to the difficulty of training a large number of parameters in the TDNN with error backpropagation, its performance suffers compared even with the linear models. However, these nonlinear models do not exhibit any significant improvement for target reaching. This will be discussed in the following chapter.

**Analysis**

**Evaluation of Training Performance for NMCLM**

Now, we demonstrate the advantage of training in NMCLM compared to the TDNN using the food reaching data. The topology proposed in NMCLM is basically equivalent to a three-layer network: the first layer of weights consists of the competitive
model coefficients, the second and third layer of weights are simply the weights of the following MLP. In this topology, the first hidden layer and the output layer have linear PEs, whereas the second hidden layer has nonlinear PEs. In the NMCLM approach, the first layer weights are trained competitively to predict the desired signal, whereas the MLP is optimized using error backpropagation.

In order to quantify the performance of this training procedure from an information-theoretic point-of-view, we evaluate the mutual information $I(z_C, d)$, between the outputs of the competitive models, $z_C$, and the desired output, $d$. Using a Parzen window estimator for the mutual information [Erd02] on ten arbitrary segments of the hand trajectory (each of length 1000 samples), the average and standard deviation of $I(z_C, d)$ is found to be 8.97 nats ($\pm$ 1.21 nats). The maximum mutual information allowed by this model and data, obtained by estimating $I(z_C, \hat{d})$ is 9.83 nats ($\pm$ 1.19 nats). Percentage-wise, the information contained in the competitive model outputs pertaining to the desired output is thus 92% ($\pm$ 6%). From this, we conclude that the information loss in the first layer is just 8% ($\pm$ 6%).

For comparison, another network with the same topology is trained as follows: The MLP weights are borrowed from the second hidden layer and the output layer of the above network (in order to ensure identical information loss at this stage). The first layer weights are then trained using standard backpropagation through these MLP weights, instead of using competition. This network, therefore, uses the minimum MSE solution for the first layer weights. Similarly, the mutual information $I(z_B, d)$ between the output of the first layer of this network $z_B$, and the desired output $d$ is calculated to be 7.42 nats ($\pm$ 1.35 nats). For this network, the maximum mutual information is 10.90 nats ($\pm$ 0.40 nats).
These correspond to an information-transfer percentage of 68% (±11%). Therefore, the information loss in the first layer of the second network is 32% (±11%).

In summary, the mutual information between the desired output and the competitive model outputs is larger than the first layer outputs of the equivalent TDNN (all the weights are trained only by error backpropagation), which shows that the training in NMCLM is more efficient.

**Analysis of Linear Filters**

It is intriguing to pose a question of what is the value of adapting the parameters in the input layer where most of the weights reside. For this, we analyze the pole zero plots of the trained FIR filters for each neuron from multiple linear models. In this analysis, we verify that there are only minor variations in the pole zero plot no matter what is the neuron or the adaptation procedure. Figure 5-3 shows the frequency responses of the 10 linear filters (with 10-tap delay line) in NMCLM for the food reaching task for a specific neuron. These frequency responses indicate that they are all lowpass filters and the locations of the zeros (denoted by different markers for different models) for all models are similar. It means that the role of the filters is to lowpass filter (smooth) the input. As depicted, the zeros tend to be placed at equal intervals very close to the unit circle. The major difference seems to be the gain at DC.

Hence, one can synthesize an alternate adaptive filter that displays a very similar response and has only two free parameters, as

$$H(z) = G \frac{1 + z^{-10}}{1 - \alpha z^{-1}}$$  \hspace{1cm} (5-5)

where the two free parameters encode the gains ($G$) and the locations of the pole of the
filter for each neuron ($\alpha$), imposing the constraint $|\alpha| < 1$. The number of NMCLM weights with this filter for the estimation of one output coordinate can then be reduced from 30,630 to 6,870. The performance evaluation of this simplified model shows a slightly low level compared to the original performance (the performance profile is similar to the Wiener filter for the prediction of movement, while superior to the linear models for rest). \(^1\) This indicates that a variable gain control and a variable integration over time per neuron seem sufficient to derive optimal models for BMIs. These characteristics can be obtained by a multitude of systems that can be much easier to implement and do not even require adaptation. Further work will be pursued along this line.

\(^1\) The numerical results are followings; CC(move)= 0.76 (± 0.18), SER(move)= 4.61 dB (± 2.31 dB), CC(rest)= 0.01 (± 0.26), SER(rest)= 7.67 dB (± 4.43 dB). See chapter 6 for the comparison of these results with others.
CHAPTER 6
COMPARISON OF MODELS

In this chapter, we summarize the evaluation of the generalization performance for all models introduced so far in this dissertation. We emphasize, however, that the comparison is done for the datasets of 100-200 simultaneously recorded neurons for which the standard Wiener filter algorithm yielded very good performance. With the increase of the number of simultaneously recorded neurons, task complexity, and complexity of predicted motor parameters, what we will see only tendencies in this comparison may become important for BMI designs.

Before presenting comparison results, we first demonstrate the outputs of every model along with the actual hand trajectories for food reaching in Fig. 6-1 and for target reaching in Fig. 6-2, respectively. Since our approaches have been developed by assigning the Wiener filter as a golden standard, observations in these figures are likely to be made mainly by comparing trajectories of models with that of the Wiener filter. First, we can observe that NLMS can predict better for rest positions than the Wiener filter in Fig. 6-1b. This explains how a time-varying learning rate in NLMS can help track a nonstationary data. Next, we can see that regularized models yield smoother output trajectories than the Wiener filter especially during rest. Also, it can be easily captured that NMCLM provides the most accurate prediction in Fig. 6-1f. NMCLM shows its ability to stay in rest position with little jitters, and to track rapid changes of hand trajectory during movements. This may be due to the nonlinear structures in NMCLM.

On the other hand, in Fig. 6-2, all models show similar prediction performance for
Figure 6-1. The actual hand trajectory (dotted red line) and the estimated hand trajectory (solid black line) in the x-, y-, and z-coordinate for the 3D food reaching task on a sample part of the test data; (a) the Wiener filter, (b) the linear filter with NLMS, (c) the subspace Wiener filter, (d) the gamma filter, (e) the linear filter regularized by weight decay, and (f) NMCLM.
Figure 6-2. The actual hand trajectory (dotted red line) and the estimated hand trajectory (solid black line) in the x-, and y-coordinate for the 2D target reaching task on a sample part of the test data. (a) the Wiener filter, (b) the linear filter with NLMS, (c) the subspace Wiener filter, (d) the gamma filter, (e) the linear filter regularized by weight decay, and (f) NMCLM.
target reaching. None of models outperforms visually in the output trajectories.
Performance measures presented later will demonstrate this similarity of performance
(although there are statistical differences between models).

**Comparison of Model Parameters**

We now compare the weights of four linear models: the Wiener filter, the linear
model trained by NLMS, the gamma filter, and the linear model regularized by weight
decay. Since the number of tap delays is different among models, the weights must be
represented based on neurons (not every tap of different time lag). Hence, we compute
the average value of the weight magnitudes over tap delays and over three (or two) output
dimensions. Then, the standard deviation of each neuronal data estimated from the
training set is multiplied by the average magnitude to obtain a measure of neuronal
contribution; that is, the average sensitivity of the output to individual neurons [San03a].

Figure 6-3 shows the calculated sensitivities in each model for both food reaching and
target reaching. Note that we rescale the sensitivity values to be in [0, 1] in order to
facilitate the visual comparison.

It can be observed in Fig. 6-3a that the normalized weight magnitude distributions are
similar among models except for the gamma filter. The weight distribution of NLMS
follows that of the Wiener filter. But, it exhibits smaller magnitudes when the
corresponding neurons do not contribute much. This may explain the regularization
property of NLMS with the constraint on the weights as presented in chapter 4. Weight
decay also prunes weights, generating the sparse weight distribution, which can enhance
generalization. The weight distribution of the gamma filter might differ from others since
it utilizes the different time scale. It weights more on neurons indexed as 57, 84, 87 and
94, where neuron 57 is the neuron with highest firing rate, and neuron 94 is one of the
highest sensitivity neurons according to the analysis in Sanchez et al. [San03b]. For the target reaching task as shown in Fig. 6-3b, all models present similar weight magnitude distributions, which may explain the similar performance of all models.

Figure 6-3. The distributions of normalized weight magnitudes of four linear models over neuronal space for; (a) food reaching, and (b) target reaching.
Performance Evaluation

Tables 6-1 and 6-2 summarize the generalization performances of all models using measures introduced in chapter 3. For food reaching, there are ten reaching movements in the test data for which the performances are measured. The CEM curves of all models are presented in Fig. 6-4. Since the CEM curve measures the probability that the distance between the estimated and actual hand positions is less than a given quantity represented

Table 6-1. The generalization performances of linear models and nonlinear models for the 3D food reaching task.

<table>
<thead>
<tr>
<th>Measures</th>
<th># of weights</th>
<th>CC (move) (dB)</th>
<th>SER (move) (dB)</th>
<th>CC (rest) (dB)</th>
<th>SER (rest) (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiener</td>
<td>2973</td>
<td>0.76 ± 0.19</td>
<td>4.76 ± 1.87</td>
<td>0.03 ± 0.22</td>
<td>2.40 ± 2.80</td>
</tr>
<tr>
<td>NLMS</td>
<td>2973</td>
<td>0.75 ± 0.20</td>
<td>4.85 ± 2.11</td>
<td>0.06 ± 0.22</td>
<td>3.40 ± 2.76</td>
</tr>
<tr>
<td>Gamma</td>
<td>1191</td>
<td>0.78 ± 0.19</td>
<td>5.25 ± 1.97</td>
<td>0.07 ± 0.21</td>
<td>3.59 ± 3.11</td>
</tr>
<tr>
<td>Subspace</td>
<td>1113</td>
<td>0.77 ± 0.18</td>
<td>4.84 ± 2.06</td>
<td>0.09 ± 0.20</td>
<td>3.78 ± 2.57</td>
</tr>
<tr>
<td>Weight decay</td>
<td>&lt;2973</td>
<td>0.77 ± 0.18</td>
<td>4.73 ± 2.04</td>
<td>0.07 ± 0.22</td>
<td>3.76 ± 2.78</td>
</tr>
<tr>
<td>Kalman</td>
<td>1017</td>
<td>0.78 ± 0.20</td>
<td>4.32 ± 1.97</td>
<td>0.05 ± 0.25</td>
<td>2.26 ± 3.85</td>
</tr>
<tr>
<td>TDNN</td>
<td>29823</td>
<td>0.77 ± 0.17</td>
<td>4.87 ± 2.56</td>
<td>0.02 ± 0.22</td>
<td>3.29 ± 5.67</td>
</tr>
<tr>
<td>NMCLM (FIR)</td>
<td>30753</td>
<td>0.81 ± 0.15</td>
<td>5.90 ± 3.00</td>
<td>0.03 ± 0.22</td>
<td>5.64 ± 4.00</td>
</tr>
<tr>
<td>NMCLM (Gamma)</td>
<td>12933</td>
<td>0.81 ± 0.19</td>
<td>6.08 ± 3.19</td>
<td>0.06 ± 0.23</td>
<td>6.23 ± 5.23</td>
</tr>
</tbody>
</table>

Table 6-2. The generalization performances of linear models and nonlinear models for the 2D target reaching task.

<table>
<thead>
<tr>
<th>Measures</th>
<th># of weights</th>
<th>CC (x) (dB)</th>
<th>SER (x) (dB)</th>
<th>CC (y) (dB)</th>
<th>SER (y) (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiener</td>
<td>3842</td>
<td>0.66 ± 0.02</td>
<td>2.42 ± 0.54</td>
<td>0.48 ± 0.10</td>
<td>1.08 ± 0.52</td>
</tr>
<tr>
<td>NLMS</td>
<td>3842</td>
<td>0.68 ± 0.03</td>
<td>2.42 ± 0.55</td>
<td>0.50 ± 0.08</td>
<td>0.90 ± 0.49</td>
</tr>
<tr>
<td>Gamma</td>
<td>3842</td>
<td>0.70 ± 0.02</td>
<td>2.81 ± 0.69</td>
<td>0.53 ± 0.09</td>
<td>1.55 ± 0.43</td>
</tr>
<tr>
<td>Subspace</td>
<td>882</td>
<td>0.70 ± 0.03</td>
<td>2.80 ± 0.83</td>
<td>0.58 ± 0.08</td>
<td>1.90 ± 0.57</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>&lt;3842</td>
<td>0.71 ± 0.03</td>
<td>2.79 ± 0.92</td>
<td>0.57 ± 0.08</td>
<td>1.75 ± 0.46</td>
</tr>
<tr>
<td>Kalman</td>
<td>1188</td>
<td>0.71 ± 0.03</td>
<td>2.77 ± 0.65</td>
<td>0.58 ± 0.10</td>
<td>1.63 ± 0.76</td>
</tr>
<tr>
<td>TDNN</td>
<td>57691</td>
<td>0.65 ± 0.03</td>
<td>2.24 ± 0.59</td>
<td>0.51 ± 0.08</td>
<td>1.10 ± 0.39</td>
</tr>
<tr>
<td>NMCLM (FIR)</td>
<td>58622</td>
<td>0.67 ± 0.03</td>
<td>2.62 ± 0.53</td>
<td>0.50 ± 0.07</td>
<td>1.23 ± 0.40</td>
</tr>
<tr>
<td>NMCLM (Gamma)</td>
<td>58622</td>
<td>0.67 ± 0.02</td>
<td>2.55 ± 0.61</td>
<td>0.47 ± 0.07</td>
<td>0.95 ± 0.40</td>
</tr>
</tbody>
</table>
at the x-axis, the closer the curve is to the upper left corner the better the corresponding model performs. To visualize the performance clearly, we give an instance of the CEM profile for certain distance; the models are listed in the order of \( Pr(|e| \leq 20 \text{mm}) \), where the top model exhibits the highest probability. Figure 6-4 shows that the differences among models are more distinguishable in the food reaching task than in the target hitting task. Also, NMCLM demonstrates superior performances for the food reaching task, while it does not improve performance for the target hitting task.

![Figure 6-4. Comparison of the CEM of the nine models for (a) the food reaching task, and (b) the target reaching task.](image)

**Statistical Performance Comparison**

To quantify the performance evaluations obtained above, we test the statistical difference between the Wiener filter and all the other models [Kim05a]. We first assume that the average magnitude of the error vector \( E[|e|] \) on the test data is a sufficient measure of model performance. To compare the performance of different models, we test the difference between the distributions of \( E[|e|] \). \( E[|e|] \) is locally estimated in individual 4-second non-overlapping time windows through the test data (approximately 3,000-second long). Since a summation is used to estimate the mean, the set of \( E[|e|] \) can be
assumed to be drawn from a Gaussian distribution based on the central limit theorem (CLT). Also, the use of non-overlapping windows can approximately satisfy the independence condition between the estimates of $E[|e|]$ from different windows. Therefore, the t-test can be applied to the set of $E[|e|]$.

In order to setup a test environment between one model with the Wiener filter, we first define $\Delta$ as the difference between $E[|e|]$ for the Wiener filter and for one of the other models,

$$\Delta(k) = E[|e|]_M(k) - E[|e|]_W(k)$$

(6-1)

where $E[|e|]_M(k)$ denotes the average magnitude of error vectors in the $k^{th}$ window for the model under comparison and $E[|e|]_W(k)$ for the Wiener filter. Note that $\Delta$ is a Gaussian random variable since the linear combination of two Gaussian variables $E[|e|]_M$ and $E[|e|]_W$ is also a Gaussian variable. Then, we apply the t-test to $\Delta$ with the realizations $\{\Delta(k)\}$. The hypotheses for the one-tail t-test then become,

$$H_0 : E[\Delta] \geq 0$$

$$H_A : E[\Delta] < 0$$

(6-2)

Given the significance level of $\alpha$, if the null hypothesis is rejected we can claim with the confidence level of $(1-\alpha)$ that the compared model performs better than the Wiener filter.

The t-test results are presented in table 6-3. For the food reaching task, every model performs better than the Wiener filter except the TDNN. Note that the TDNN shows higher level of mean SER during rest, but with a relatively large variance. For the target reaching task, however, only three linear models pruned by regularization are shown to
outperform the Wiener filter. These results are fairly consistent with results in tables 6-1, 6-2, and Fig. 6-4.

Table 6-3. The t-test results for the difference of the magnitude of error vectors from the test dataset between the Wiener filter and other models.

<table>
<thead>
<tr>
<th></th>
<th>Food reaching</th>
<th>Target Reaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significance level</td>
<td>0.01</td>
<td>0.05</td>
</tr>
<tr>
<td>NLMS</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gamma</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Subspace</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Weight Decay</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Kalman</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TDNN</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NMCLM (FIR)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NMCLM (Gamma)</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\(^1\)The test result of 0 indicates the acceptance of null hypothesis; while 1 indicates the rejection of null hypothesis.
CHAPTER 7
MULTIRESOLUTION ANALYSIS FOR BMI

Most designs of decoding algorithms for BMIs including our models have used the estimate of the local firing rate of neurons that is estimated by binning neural spikes with a non-overlapping sliding time window of the length ranging from 50ms up to 100ms [Cha99, Ser02, Tay02 and Wes00]. These representations of the firing rate have been used for modeling of the relationship with responsive motor parameters. Adaptive models (including linear and nonlinear ones) based on this estimate have predicted motor parameters with a correlation coefficient between 0.6 ~ 0.8. However, it has also been shown in previous chapter that all the models reached the same basic performance level especially for the target reaching task, which may not be sufficient for more involved real applications.

These results lead us to revisit our approaches for designing decoding models; extracting advanced features from neural data followed by developing an adequate mathematical decoding algorithms and topology may bring us a better decoding model. Extracting desirable features in complex, high-dimensional neuronal data is, though, an open problem, requiring intensive studies. Yet, we present here a simple approach by considering the representational space for neuronal firing activity, which will demonstrate how extracting features in input can help to improve the model performance.

In our approach, we revise the present representation of a local firing rate, encoded in a series of bin counts within a fixed width time window. Since a local firing rate can represent the local frequency of a neural spike train, the features can be extracted based
on local frequency. One of the methods for the representation of local frequency information is the multiresolution analysis [Mur04], usually realized in wavelet transform. With the multiresolution analysis, it is possible to represent the time-frequency characteristics of a signal. Basically, we can obtain as many local frequency components as we want at a given time instance. Hence, the multiresolution analysis of neural spikes may provide richer information about neuronal behavior compared to the binning using a fixed width time window.

If we consider the multiresolution analysis for spike trains, it is easy to see that binning process is nothing but discrete wavelet transform (DWT) using a Haar wavelet [Dau92]. However, since the original DWT is basically a non-causal process, a wavelet transform featuring causality should be considered. For this purpose, we employ the à trous wavelet transform [She92] to implement a causal DWT. With this procedure, the multiresolution analysis for spike trains can be regarded as binning spike trains with the multi-scale windows. Hence, the decoding models, which have been designed upon the bin count data, need not be fundamentally modified to the multiresolution data.

With the multiresolution data, however, the regularization of decoding models must be considered due to the increased input dimensionality and the collinearity between input channels. Among the number of regularization techniques used in data mining, the method based on the $L_1$-norm penalty will be more suitable since it is able to generate a sparser model than others using the $L_2$-norm penalty. It also enables us to understand the association of neuronal activities with behavior, by selecting more correlated channels.

Similar works for the multiresolution analysis for neural spike trains have been done in various research groups. Lee has estimated the cross-spectrum using wavelet
analysis between simultaneously recorded spike trains, revealing the phase-locked oscillation between spike trains [Lee02]. Laubach has demonstrated the wavelet-based processing of spike trains from the motor cortex of a behaving rat [Lau04]. He utilized discriminant pursuit (DP) [Buc95], which is based on wavelet analysis, to improve the discriminant analysis methods for better statistical predictions of temporally localized events. Cao has worked on the Haar wavelet analyses of spike trains to understand the characteristics of spike trains and enhance the decoding models in neural prosthetic systems [Cao03]. This work seems to be most relevant to our approach presented here. However, one of the major differences is that he pruned wavelet coefficients by using information theoretic measures (e.g., the mutual information) between each neuron and behavior, followed by building decoding models (e.g., Bayesian classifiers) with those pruned coefficients. On the other hand, we contain all wavelet coefficients in the input channel of the linear model and prune inputs by a regularization technique. Therefore, in our approach we can select wavelet coefficients which explicitly contribute to the output of the designated model architecture, while the selected coefficients by the mutual information method may not directly contribute to the specific decoding model. We also propose to use the à trous wavelet transform instead of standard DWT to link the multiresolution analysis with binning process for real-time applications, which has not been explicitly shown in Cao’s works.

In this chapter, we design a linear model with the multiresolution input data for BMIs, which is learned by the regularization method based on the $L_1$-norm penalty. The multiresolution input for each neuron is composed of the instantaneous spike count binned by multiple time windows of various widths. We investigate the trained linear
model using the multiresolution input for the analysis of neuronal firing activities. Next, a comparison of the multiresolution based model with the single resolution model is demonstrated. For this comparison, each channel of the multiresolution input is embedded by a time delay line in the same way as the single resolution model is formed (see Figure 3-1 for this structure). The performances of two models are evaluated. Finally, a combination of linear and nonlinear networks is considered to investigate the possibility of the performance improvement over linear models. With the optimally designed linear model using the multiresolution input, an additional nonlinear network is added in order to further reduce residuals from the learned linear model. This approach will help us to understand how much a nonlinear structure can help to a linear model when we utilize the multiresolution input.

We would like to remark here that the data used in this chapter are collected from Aurora, which is different from previous chapters.

**Multiresolution Analysis of Neuronal Spike Trains**

An overall procedure of the multiresolution analysis in BMIs is as follows: The multiresolution analysis based on the Haar wavelet is applied to spike trains of 185 neurons recorded in the cortical areas of a Rhesus monkey (Aurora); see chapter 2 for data descriptions. The Haar à trous wavelet transform [Zhe99] is utilized to perform the multiresolution analysis for individual spike trains. The resulted wavelet coefficients (or equivalently, the multi-scale bin count data) are used as the input data to a linear model. The linear model is learned by a regularization method to predict the hand trajectories. The analysis of model parameters is performed to investigate the association of single neurons with target reaching movements.
Multiresolution Analysis

The multiresolution analysis of a neural spike train can be performed via the wavelet transform. To facilitate the computation, we apply the discrete wavelet transform with the dyadic Haar wavelets. This dyadic Haar wavelet is basically utilized in the à trous wavelet transform which can be implemented very effectively in hardware.

The Haar wavelet transform is the simplest form of wavelet and was introduced in the earliest development of wavelet transform [Dau92]. Here, we only introduce the functional form of the Haar wavelets. Details in the Haar wavelet transform can be found in [Dau92]. Let us first define the Haar scaling function as,

\[
\phi(x) = \begin{cases} 
1 & \text{if } x \in [0,1) \\
0 & \text{otherwise}
\end{cases}
\] (7-1)

Let \( V_j \) be the set of functions of the form,

\[
\sum_k a_k \phi(2^j x - k)
\] (7-2)

where \( a_k \) is a real number and \( k \) belongs to the integer set. \( a_k \) is nonzero for only a finite set of \( k \). \( V_j \) is the set of all piecewise constant functions whose supports are finite, where discontinuities between these functions belong to a set,

\[
\left\{ \cdots, \frac{2}{2^j}, \frac{1}{2^j}, 0, \frac{1}{2^j}, \frac{2}{2^j}, \cdots \right\}.
\] (7-3)

Note that \( V_0 \subset V_1 \subset V_2 \subset \cdots \). The Haar wavelet function \( \psi \) is defined by,

\[
\psi(x) = \phi(2x) - \phi(2x - 1).
\] (7-4)

If we define \( W_j \) as the set of functions of the form

\[
\sum_k a_k \psi(2^j x - k),
\] (7-5)

then, it follows that
\[ V_j = W_{j-1} \oplus W_{j-1} \oplus \cdots \oplus W_0 \oplus V_0 \quad (7-6) \]

where \( \oplus \) denotes the union of two orthogonal sets.

The discrete wavelet transform (DWT) using a dyadic scaling is often used due to its practical effectiveness. The output of the DWT traditionally forms a triangle to represent all resolution scales. This form is resulted from decimation (holding one sample out of every two), and has the advantage of reduction in computational complexity and storage. However, it is not possible to obtain representation with different scales at every time instance with the decimated output. This problem can be overcome by a non-decimated DWT [Aus98] which requires more computations and storage. The non-decimated DWT can be formed in two ways; 1) the successive resolutions are obtained by the convolution between a given signal and an incremental dilated wavelet function. 2) the successive resolutions are formed by smoothing with an incremental dilated scaling function, and taking difference between successive smoothed data.

The à trous wavelet transform follows the latter procedure to produce a multiresolution representation of the data. In this transform, successive convolutions with a discrete filter \( h \) is performed as

\[ v_{j+1}(k) = \sum_{l=-\infty}^{\infty} h(l)v_j(k + 2^j l), \quad (7-7) \]

where, \( v_0(k) = x(k) \), the original discrete-time series. In its first introduction [She92], the filter \( h \) was defined as a B3 spline; (1/16, 1/4, 3/8, 1/4, 1/16). Then, the difference between successive smoothed outputs is computed as

\[ w_j(k) = v_{j-1}(k) - v_j(k) \quad (7-8) \]
where $w_j$ represents the wavelet coefficients. It is clear that the original time series $x(k)$ can be decomposed as

$$x(k) = v_S(k) + \sum_{j=1}^{S} w_j(k)$$

(7-9)

with $S$ being the number of scales. The computational complexity of this algorithm is $O(N)$ for the data length $N$.

Note that the à trous wavelet transform does not account for a causal time series where the future data are not available in the present computation of wavelet transform. To apply the à trous wavelet transform for such a case, the Haar à trous wavelet transform can be used [Zhe99]. The Haar à trous wavelet transform can be regarded as the merge of the non-decimated DWT (by the à trous wavelet transform) with the Haar wavelet transform. A difference in the Haar à trous wavelet transform from the original à trous wavelet transform is that $h$ is now replaced by the filter with $(1/2, 1/2)$. For a given discrete-time series $x(k) (= v_0(k))$, the first resolution is obtained by convolution $v_0(k)$ with $h$ such that

$$v_1(k) = \frac{1}{2}(v_0(k) + v_0(k - 1)).$$

(7-10)

And the wavelet coefficients are obtained by

$$w_1(k) = v_0(k) - v_1(k).$$

(7-11)

For the $j$th resolutions,

$$v_j(k) = \frac{1}{2}(v_{j-1}(k) + v_{j-1}(k - 2^{j-1}))$$

(7-12)

$$w_j(k) = v_{j-1}(k) - v_j(k)$$

(7-13)
Hence, the computation in this wavelet transform at time $k$ involves only information at $k-l$ with $l$ being a nonnegative integer.

The Haar à trous wavelet transform can provide a set of features from the time series data. One possible feature set can be extracted from the decomposition described in (7-9), where the wavelet coefficients $\{w_1(k), \ldots, w_{S-1}(k)\}$ and the last convolution output $v_{S-1}(k)$ are selected. However, if we seek to associate the Haar à trous wavelet transform with the binning process for spike trains, the set $\{v_0(k), \ldots, v_{S-1}(k)\}$ can be translated into the bin count data with multiple bin widths. To yield the multi-scale bin count data using (7-10), we only have to multiply $v_j(k)$ by $2^j$ such that

$$u_j(k) = 2^j v_j(k), \text{ for } j = 0, \ldots, S-1.$$  \hspace{1cm} (7-14)

Hence, the convolution output in the Haar à trous wavelet transform can provide the feature set related with binning. In the following models for BMIs, we will utilize the scaled convolution outputs $\{u_j(k)\}$ for $j = 0, \ldots, S-1$, or equivalently, the bin count data with different widths, as the input features.

**Multiresolution Analysis for the BMI Data**

In order to apply the multiresolution analysis to the BMI data, we must choose the suitable set of scales. Although it is not straightforward to determine a set of scales for the Haar à trous wavelet transform of spike trains, we may take the characteristics of neuronal data collected from our BMI paradigm into consideration for the determination of scales. Basically, the smallest scale must be larger than 1ms because of the refractory period of neuronal firing. Also, the largest scale may not exceed 1sec since it has been reported that the past neuronal activity up to 1 second is correlated with the current
movement [Wes00]. In our experiments, we select eight scales starting at 5ms up to 640ms with the dyadic scaling; 5, 10, 20, 40, 80, 160, 320, and 640ms\(^1\).

With the selected scales, the Haar à trous wavelet transform is performed on each neuronal spike train in Aurora’s dataset. Instead of performing the wavelet transform directly on raw spike trains, we first generate the basic bin count data with a 5ms non-overlapping window for every neuronal channel. Next, the Haar à trous wavelet transform is applied to the 5ms bin count data at each neuronal channel, yielding the convolution output \(v_j(k)\) for \(j = 0, \ldots, 7\) following the equation (7-12). Each series \(v_j(k)\) is then multiplied by \(2^j\) to generate \(u_j(k)\). An illustrative example of the generated \(u_j(k)\) at specific time instance \(k_0\) is presented in Fig. 7-1.

![Figure 7-1](image)

Figure 7-1. An illustration of the scaled convolution output from the Haar à trous wavelet transform; \(u_j(k)\) for a given spike train at a time instance \(k_0\). The number in each box denotes the value of \(u_j(k_0)\) for \(j = 0, \ldots, 7\).

Note that the sampling rate in \(u_j(k)\) is 200Hz for any \(j\). In terms of a binning process, \(u_j(k)\) can be interpreted as the bin count data for a given spike train with a \(5 \times 2^j\) ms

---

\(^1\) The minimum 5ms scale is chosen by empirical observation such that the bin count data is significantly different from raw spike trains containing 1’s and 0’s. However, it must be remarked that a more rigorous procedure of choosing the minimum scale may be necessary in the future study.
time window that slides over time by step of 5ms. Therefore, $u_j(k)$ with a larger $j$ will contain more overlaps between successive bins, $u_j(k)$ and $u_j(k-1)$. Such overlaps will then yield the smoother temporal patterns of $u_j(k)$ with larger $j$.

The top panel in Fig. 7-2 demonstrates an example of $u_j(k)$ of a specific neuron for 5-second period. $u_j(k)$ for each $j$ is normalized to have the maximum value of 1. Darker pixels denote larger values. The set of $u_j(k)$ are temporally aligned with the associated hand trajectories plotted in the bottom panel. In order to view the correlation of $u_j(k)$ with the movement for each $j$, $u_j(k)$ is separately plotted on top of the hand trajectory (the x-coordinate) in Fig. 7-3 (both $u_j(k)$ and the hand trajectory are scaled to be in the similar dynamic range for visualization purpose). It demonstrates that $u_j(k)$ with larger $j$ is more

![Figure 7-2](image.png)

Figure 7-2. An example of the series of $u_j(k)$ along with the corresponding hand trajectories; (top) a matrix of $u_j(k)$ in which each row represent the scale $j$ for $j = 0, \ldots, 7$ (i.e. 5ms ~ 640ms bin widths), and the columns represent time indices over 5-second duration. (bottom) the trajectories of hand position and velocity at x-(solid), and y-(dotted) coordinates.
correlated with the hand trajectory than smaller \( j \).

![Figure 7-3](image)

Figure 7-3. The demonstration of the relation between the neuronal firing activity representation at each scale (solid lines) and the hand position trajectory at x-coordinate (dotted lines).

**The Analysis of the Linear Model Based on the Multiresolution Representation**

For the further investigation of the relationship between the multiresolution representation of neuronal firing activities and target reaching movements, we develop a linear model using \( u_j(k) \) as inputs. A discrete-time series \( u_j(k) \) for each \( j \) is normalized to have zero-mean and the unit maximum magnitude such that a model can avoid biasing to the larger-scale inputs. 185 neurons with 8 scales yielded the input dimension of 1480. The multiresolution representation for the 320-sec training dataset (containing \( 320 \times 200 = 64,000 \) samples) generates an input data matrix \( X \) (\( 64,000 \times 1,480 \)), where each row represents the input feature vector at a given time instance. Then, a linear model is designed to predict the desired response (the x-, or y-coordinates of hand position or velocity) vector \( d \) (\( 64,000 \times 1 \)) with the linear combination of \( X \) such that
\[ \mathbf{d} = \hat{\mathbf{d}} + \mathbf{e} = \mathbf{Xw} + \mathbf{e} \]  

(7-15)

where \( \mathbf{w} \) is the model weight vector and \( \mathbf{e} \) is the error vector. Note that the desired responses are normalized to have zero-mean so that the estimation of the y-intercept is not necessary.

Learning the model weight vector \( \mathbf{w} \) can be achieved by a variety of methods. However, we must consider regularization in this model due to the very high input dimensionality (>1,000). We have introduced several regularization methods in chapter 4. Among those techniques, the \( L_1 \)-norm based algorithm may be suitable since it generates a sparser model and enables the selection of input variables, which is useful for the analysis in neuronal population. Here, we utilize the LAR algorithm which learns \( \mathbf{w} \) by the stagewise selection of input variables with constraints on the \( L_1 \)-norm of \( \mathbf{w} \). Recall that this algorithm is based on the assumption that the input channels (or columns in \( \mathbf{X} \)) are not linearly dependent to each other\(^2\).

To determine the threshold for the \( L_1 \)-norm of weight vector in the LAR algorithm, we utilize the hold-out cross-validation. We hold out the last 10% of the training data as the validation set. The threshold is determined by minimizing the MSE for the validation set. The LAR algorithm stops learning when the \( L_1 \)-norm reaches this threshold.

The LAR algorithm selects a different subset of input channels for each desired response (there are four responses including the \( x \)-, and \( y \)-coordinates of hand position and velocity). From the trained weight vectors, we select neurons that have nonzero weights for at least one scale (recall that there are eight scales per neuron). Then, we

\(^2\) Although more thorough analysis must be executed, we can empirically test if the rank of \( \mathbf{X} \) is equal to the number of channels. And, the empirical results shows that at least for the matrix \( \mathbf{X} \) used in this study, input channels are not linearly dependent.
examine the distribution of the selected neurons over multiple cortical areas. The number of selected neurons and its portion for each area are shown in table 7-1 (see table 2-1 for the description of cortical areas in Aurora’s dataset). In this table, we can observe that more neurons are selected in the case of predicting velocity. Although the biological analysis of this result must be complemented, it might be caused by the fact that the trajectory of velocity changes more rapidly than that of position, thus requiring finer resolution inputs.

Table 7-1. The number of the selected neurons in each cortical area.

<table>
<thead>
<tr>
<th></th>
<th>PMd</th>
<th>M1</th>
<th>S1</th>
<th>SMA</th>
<th>M1_ipsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position-x</td>
<td>18 (27%)</td>
<td>27 (47%)</td>
<td>9 (24%)</td>
<td>7 (37%)</td>
<td>2 (40%)</td>
</tr>
<tr>
<td>Position-y</td>
<td>20 (30%)</td>
<td>26 (39%)</td>
<td>15 (39%)</td>
<td>7 (37%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td>Velocity-x</td>
<td>50 (76%)</td>
<td>46 (81%)</td>
<td>30 (79%)</td>
<td>15 (79%)</td>
<td>5 (100%)</td>
</tr>
<tr>
<td>Velocity-y</td>
<td>40 (61%)</td>
<td>42 (74%)</td>
<td>30 (79%)</td>
<td>13 (79%)</td>
<td>3 (60%)</td>
</tr>
</tbody>
</table>

Figure 7-4 describes the selection results for each desired response. The black pixels denote the selected variables aligned in neuronal space (x-axis) with the scales in the y-axis. These graphs show that LAR prefers selecting inputs with larger scales since the temporal trajectories of larger scales exhibit more correlation with movement as shown in Fig. 7-3.

**Comparison of Models with the Multiresolution Representation**

We now seek to answer the following questions;

- Can the multiresolution representation of the neuronal firing activity improve the prediction performance of decoding models for BMIs compared to the single resolution representation?

- If so, how much does it improve performance?

Two linear models are designed with different input datasets; the first model receives the single resolution data, i.e., the bin count data with a fixed width window of

---

3 The ratio of the number of selected neurons to the total number of neurons.
Figure 7-4. The distribution of the selected input variables for (a) x-coordinate, (b) and y-coordinate of position, and (c) x-coordinate, and (d) y-coordinate of velocity.

80ms as inputs and the second model receives the multiresolution data with eight resolution levels (scales) from 5ms up to 640ms\(^4\). Normalization and embedding are applied to every channel in both inputs (single resolution and multiresolution input data); each input channel is normalized to have zero-mean and the unit maximum magnitude and a 6-tap time delay line is used to embed the bin count data at each channel. This embedding results in an 1110 (6×185) dimensional input space for the single resolution model and an 8880 dimensional input space for the multiresolution model, respectively. The same training dataset as above (320-sec data) is used for both models. However, the number of training samples is different between models since two input data are binned with different windows: the single resolution data are generated by binning with a 80ms non-overlapping window, yielding 4,000 samples for 320 seconds, and the

\(^4\) The 80ms bin width is chosen since it belongs to a set of scales. It means that the single resolution representation can be a special case of multiresolution representation using only one scale.
multiresolution data are generated at 200Hz rate, yielding 64,000 samples. Hence, the
first model uses the desired signals sampled at 12.5Hz, and the second model uses the
tones sampled at 200Hz, respectively. Both models are trained to predict 2D hand position
and velocity by the LAR algorithm. The threshold for the $L_1$-norm of the weights in the
LAR algorithm is determined by the hold-out cross-validation.

The number of nonzero weights after training is listed in table 7-2. It is noteworthy
that the weights for the multiresolution data are more pruned compared to single
resolution. It may indicate that there is redundancy between large-scale inputs and small-
scale inputs. So, the LAR algorithm, which exploits the correlation of inputs with the
hand trajectories, is inclined to select a large-scale input to preserve large correlation at
the cost of losing temporal resolution. On the other hand, the LAR algorithm with single
resolution inputs may need much more inputs to reduce the correlation of the selected
inputs with residuals.

Table 7-2. The number of the nonzero weights.

<table>
<thead>
<tr>
<th></th>
<th>Position-x</th>
<th>Position-y</th>
<th>Velocity-x</th>
<th>Velocity-y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single resolution</td>
<td>568 (51.1%)</td>
<td>675 (60.8%)</td>
<td>359 (32.3%)</td>
<td>379 (34.1%)</td>
</tr>
<tr>
<td>Multiresolution</td>
<td>344 (3.9%)</td>
<td>449 (5.1%)</td>
<td>379 (4.3%)</td>
<td>580 (6.5%)</td>
</tr>
</tbody>
</table>

Next, we examine which neurons are selected by LAR in both models. We collect
neurons which are assigned with at least one nonzero weight by LAR for entire time lags
(and scales for the multiresolution model). In table 7-3, the number of the selected
neurons in each model and the number of neurons which are selected in both models are
depicted for four different desired responses. The last row in the table represents the
number of neurons which are commonly selected for both models. We can see in this
table that neuronal subsets selected for both models are very similar. It may indicate that

---

5 The ratio of the nonzero weights to the total number of weights.
the multiresolution representation may not change the linear model to exploit the
different input information from the case of the single resolution bin data.

Table 7-3. The number of neurons selected by LAR for each models.

<table>
<thead>
<tr>
<th></th>
<th>Position-x</th>
<th>Position-y</th>
<th>Velocity-x</th>
<th>Velocity-y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single resolution</td>
<td>143</td>
<td>159</td>
<td>121</td>
<td>122</td>
</tr>
<tr>
<td>Multiresolution</td>
<td>117</td>
<td>144</td>
<td>124</td>
<td>160</td>
</tr>
<tr>
<td>Common</td>
<td>107</td>
<td>138</td>
<td>105</td>
<td>116</td>
</tr>
</tbody>
</table>

We use three performance measures introduced in chapter 3 for both model outputs; correlation coefficients (CC), signal-to-error power ratio (SER), and the cumulative error metric (CEM). These measures are evaluated in the test dataset to assess generalization performance as summarized in table 7-4. The evaluation reveals the superior performance of the linear model with multiresolution data compared to the one with the single resolution data. The CEM curves are also shown in Fig. 7-5, by which we can observe that a probability that the length of the error vector is less than certain positive number is higher for the multiresolution model than with the single resolution model. However, it is also notable that the increase of performance due to the multiresolution representation is rather marginal. To assess the statistical difference of performances between two models, we perform the t-test based on MSE following the procedure presented in chapter 6. The null hypothesis of no difference between model

Table 7-4. Performance comparison between the multiresolution and the single resolution models.

<table>
<thead>
<tr>
<th>Measures</th>
<th>Single Resolution</th>
<th>Multiresolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CC</td>
<td>SER</td>
</tr>
<tr>
<td>Position-x</td>
<td>0.73±0.02</td>
<td>3.37±0.73</td>
</tr>
<tr>
<td>Position-y</td>
<td>0.68±0.06</td>
<td>2.00±0.60</td>
</tr>
<tr>
<td>Velocity-x</td>
<td>0.71±0.03</td>
<td>2.80±0.82</td>
</tr>
<tr>
<td>Velocity-y</td>
<td>0.76±0.02</td>
<td>3.78±0.39</td>
</tr>
</tbody>
</table>
Figure 7-5. The CEM curves of the single resolution model (red dotted lines), and the multiresolution model (black solid lines); (a) for hand position and (b) for hand velocity.

performances is rejected at significance level of 0.05/0.01 (p<0.001) for both position and velocity, statistically proving a superior performance of the multiresolution model.

**Combination of Linear and Nonlinear Models**

We have observed that the neuronal firing activity features extracted by the multiresolution analysis could improve prediction performance of a linear decoding model, learned by a regularization method. However, the margin of this improvement over the model receiving the single resolution input is slight. Although there are plenty of possibilities to design better linear models, we rather opt for adding supplementary nonlinear structures to the linear model for the following reasons;

- We can preserve the trained parameters in linear models such that the model is improved upon the linear model.
- A nonlinear structure may be able to predict some parts of trajectory which the linear model is unable to track.
- It can provide an opportunity to answer a general question; can nonlinear models help improve performance over linear models for BMIs?
This approach of combining nonlinear models to a linear model is motivated by the cascade correlation network proposed by Fahlman [Fah91]. This network was originally developed in order to solve the problems of MLP such as; how to optimize the number of hidden PEs and how to avoid the “herd” effect of hidden PEs. In the cascade correlation network, a linear mapper between input and output is first learned. Then, the weights of the trained linear mapper freeze and a new hidden nonlinear PE is added to a linear network. The weights on connections from inputs to the added PE are trained to maximize the correlation between PE’s output and the residual from the linear mapper. After adapting these weights, the PE is treated as a new input node, and the linear mapper including all existing inputs and this PE is learned again. Then, all the weights are fixed again, and the next PE is cascaded to the first PE. The weights from all inputs and the first PE to the newly added PE are adjusted again with the same criterion. Then, the second PE is added to the linear mapper to reconfigure weights of the linear network all over again. This procedure is continued until some stopping criterion is met.

One of the features in cascade correlation is that it sets a basis on the linear mapper and adds up nonlinear PEs to explain the residual of desired response that the linear network cannot predict by itself. This can be viewed as a sequential construction of a set of basis where an initial set of inputs compose a basic subset and a set of nonlinear PEs form additional cascade nonlinear bases. Therefore, the cascade correlation can provide a framework in which we can examine if there is room for improvement after fitting a linear model using nonlinear bases.

However, this architecture requires quite a number of computations due to its repetitive learning of linear connections from entire inputs and additional PEs to output,
especially when the input dimensionality is very high. Also, the learning technique must be delicately designed for linear connections with additional PEs when generalization is a special issue. Therefore, the direct application of cascade correlation may not be suitable to BMI modeling based on the multiresolution data.

As an alternative, we choose to add a nonlinear neural network instead of nonlinear PEs to explain the residual from the linear mapper at once. This is motivated by the empirical observation that single nonlinear PE (sigmoid nonlinearity is used here) can hardly yield the output from a large number of neuronal inputs which is significantly correlated with the residual. Although a nonlinear network can bring many modeling issues such as topology, learning methods, and the network size, it can reduce the computational burden of learning, while still demonstrating whether nonlinear models improve performance or not.

**Nonlinear Modeling**

Among a number of approaches to build neural networks, we need to determine which model suits better to our environment. Main decision factors can be listed as the ability to handle the high input dimensionality, the low computational complexity, and a suitable nonlinearity for the explanation of the residual for the BMI data. There are two major approaches for the global function approximation using neural networks; multilayer perceptron (MLP) and radial basis functions (RBF) [Hay96b]. Major differences between two approaches are the nonlinearity in bases and training procedure; MLP usually utilizes the sigmoid nonlinear functions while RBF utilizes a radial basis function such as a Gaussian kernel. Also, the weights in all layers of MLP are trained simultaneously by a learning algorithm such as error backpropagation. But, the weights in RBF are trained sequentially such that the first layer weights are adjusted as centers for
clusters in the input space and then the second layer weights are trained using the algorithms for linear regression. Hence, once the centers for each basis are determined, the learning of the second layer weights becomes a linear regression problem which is relatively simpler than training entire network by error backpropagation through the nonlinearities in MLP.

Also, the shape of nonlinearity in RBF may be more favorable to explain the residual from the linear mapper than MLP if we look at residual. Figure 7-6 shows an example trajectory of the residual generated by the linear model designed in previous section using the embedded multiresolution neuronal data for the prediction of hand position. As we can see, the residual trajectory is similar to a sinusoidal and it is rather smooth (although we expect the residual is close to white noise, it is actually not). This smooth trajectory may be caused by the fact that the regularized linear model yields a smooth output trajectory that tends to track the low-frequency components of the hand trajectory. So, the linear model is inclined to miss high frequency components including a lot of peaks and momentary changes in movements. This fact results in a sinusoidal-like residual trajectory since it is largely impacted by peaks.

In order to estimate this residual trajectory by linearly combining nonlinear bases, the radial basis which forms peaky bell-shape nonlinearity may be more suitable. Also, the preliminary results showing that nonlinear models using sigmoid nonlinear functions (e.g. NMCLM, TDNN) for prediction in a 2D target reaching BMI performs similar to linear models may lead us to alter our choice of nonlinear functions\(^6\).

\(^6\) We empirically evaluated the generalization performances of MLP and RBF for the estimation of the residual from the linear model, and RBF exhibited slightly better performance (although, these results were not produced from the optimized models).
Simulations

Since the first layer of RBF is learned to find clusters in the input space, the high dimensionality of input may impair localization of clustering, thus yielding improper centers for radial bases (note that the input dimensionality used in the linear model is 8880). Even if we utilize only the selected inputs by LAR, there are still 684 input variables. To reduce the input dimensionality further, we select a subset of the selected inputs which are assigned with relatively large weights. Also, only the instantaneous multiresolution inputs are considered (without time delays). Sorting inputs by the magnitude of weights, we select the ones with the largest magnitudes the sum of which is approximately 90% of the sum of total magnitudes. This process selects 140 input variables.

The centers of basis with these input data are learned by a simple clustering $k$-means algorithm [Bis95]. The kernel width for every basis functions are equally set to 0.8 with which the output of basis functions exhibits smooth trajectories varying at a similar rate to that of residual. The number of basis functions is empirically determined to be
400. Note that the smaller number of basis functions will not be sufficient for estimating residual, and the larger number will suffer poor generalization. Then, the second layer weights are learned with the least squares method (including a bias term) to predict 2D residual signals.

After training an RBF network, its outputs with novel test data are added to those of the linear model. Then, its generalization performance is evaluated using the same measures as in table 7-4. A comparison of performance measures of the combinatory outputs from the linear models and RBF with the linear model only is presented in table 7-5. It shows superior performance of the combinatory network. A statistical test proposed in the previous chapter results in the rejection of null hypothesis of no significant difference between performances of two models using MSE measures, with \( p = 0.002 \).

<table>
<thead>
<tr>
<th>Measures</th>
<th>Single Resolution</th>
<th>Multiresolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CC</td>
<td>SER</td>
</tr>
<tr>
<td>Position-x</td>
<td>0.78±0.03</td>
<td>4.11±0.84</td>
</tr>
<tr>
<td>Position-y</td>
<td>0.71±0.05</td>
<td>2.33±0.67</td>
</tr>
</tbody>
</table>

Figure 7-7 demonstrates comparison of outputs from the combinatory network and the single linear model. It shows example trajectories of both models along with actual hand trajectory, for x-coordinate. We can see that the combinatory network can track peaks of hand trajectory slightly better than the linear model. It indicates that an additional nonlinear network can help reach peaks more accurately.

This experiment demonstrates that it is possible to improve further over optimally designed linear model, by employing nonlinear structures. Although we present here
simple examples showing the slight improvement of performance in a target reaching BMI (without optimization), finer designs of nonlinear networks may be able to increase performance further.

Figure 7-7. An example of the output trajectories of the combinatory network and single linear model.

Discussions

Although it has been demonstrated that the multiresolution representation of neuronal spikes enables a better decoding model for BMIs, the consequent performance improvement is marginal. Even with a more sophisticated model which combines the linear model and the nonlinear neural networks, prediction performance is still far from the desirable level for practical use of BMIs. It means that there are still a lot of extents for performance improvement.

However, it does not mean that the multiresolution analysis study is not useful in BMI modeling. In fact, it reveals the relationship between neuronal firing rates and the associated behavior for individual neurons. Also, the increased performance by the
multiresolution input data over the single resolution (fixed width bin) data using exactly the same linear model may inspire us to consider data mining of neuronal firing activities in order to design more accurate BMI decoding models.

It is quite interesting to view multiresolution analysis using the Haar à trous wavelet transform in the frame of the generalized feedforward filter [Pri93]. The relationship between the generalized feedforward filter and the wavelet transform was analyzed in [Che98] where the continuous wavelet transform was implemented by the Laguerre filter. This study showed that the difference of the adjacent tap outputs could implement the wavelet decomposition of the input signal. As introduced in chapter 4, in the generalized feedforward filter, an instantaneous input signal is delayed by a delay operator of \( G(z) \). The gamma delay operator used in the gamma filter is given by,

\[
G(z) = \frac{\mu}{z - (1 - \mu)} \tag{7-16}
\]

where \( \mu \) is a feedback parameter. On the other hand, a delay operator induced by the Haar à trous DWT represented in (7-12) can be given by,

\[
H_k(z) = 0.5 + 0.5z^{-2^k-1} \tag{7-17}
\]

for the \( k \)th tap. Note that \( G(z) \) is constant for all taps while \( H_k(z) \) varies its order over taps.

The other distinction is the fact that \( G(z) \) has an IIR structure whereas \( H_k(z) \) forms an FIR filter for \( k \). The transfer function from an input to the \( k \)th tap output for the gamma filter, \( G_k(z) \) is given in (4-7). The transfer function from an input to the \( k \)th tap for the Haar à trous DWT is obviously an moving average filter of an order of \( 2^{k-1} \) with gain of 0.5. Hence, if we set \( 0 < \mu < 1 \) for the gamma filter, both transfer functions feature lowpass filtering.
Figure 7-8 shows the outputs from four taps of two generalized feedforward filters with \( G(z) \) and \( H_a(z) \), respectively. The input signal to filters is bin count with 80ms time window of a specific neuronal activity in Aurora dataset. These tap outputs are plotted along with x-coordinate of actual hand trajectory. \( \mu \) for the gamma delay is set to 0.5. In order to view temporal patterns of each tap output, we divide each output by its maximum absolute value, as done in previous analysis in this chapter.

In this figure, it can be easily seen that two filters produce similar tap outputs for given input except the different memory depths. This reveals that the multiresolution analysis based on the Haar \( \text{à trous} \) DWT generates a feature space of input that is quite similar to the space generated by the gamma filter. This is evident by the relationship between the generalized feedforward filter and the wavelet transform. Recall that the wavelet coefficient in the Haar wavelets is obtained by subtracting the current
convolution output from the preceding output, as depicted in (7-13). As mentioned above, the wavelet coefficients can be obtained by the difference between the consecutive tap outputs of the generalized feedforward filter. Note that the quantity of the Haar wavelet transform we have used here is not wavelet coefficients but the set of convolution output (for equalization with bin count). And this set of convolution output can be linked with the tap outputs of the generalized feedforward filter. Therefore, the multiresolution analysis outputs using the Haar wavelet transform may contain similar information to the tap outputs of the gamma filter.
CHAPTER 8
DETERMINATION OF NEURONAL FIRING PATTERNS USING NON-NEGATIVE MATRIX FACTORIZATION

With a collection of neuronal electrical activities over several cortical areas of a primate synchronously recorded with motor parameters (e.g. hand positions) during the primate’s performance of a particular movement task, it becomes more plausible to analyze a variety of aspects of the neuronal population such as its function related with behavior, the spatio-temporal structure of the population activity, and the relation of individual neurons in each cortical area to a particular motion, if we list a few. With the bin count data estimating a local neuronal firing rate and the synchronously recorded hand positions, several BMI models (e.g. the Wiener filter, recursive multilayer perceptrons, etc.) have estimated the linear or nonlinear mappings between neuronal population and behavior, as shown in previous chapters. From these models, we can extract the information about the neuronal contributions to movement.

Recently, the sensitivity of neurons and cortical areas based on their role in the mapping learned by the RMLP or the Wiener filter has been investigated [San03a]. This sensitivity analysis examined how each neuron contributes to the output of the models, and found consistent relationships between cortical regions and segments of the hand trajectory. For instance in a food reaching task, this analysis indicated that during each reaching action, specific neurons from the posterior parietal, the premotor dorsal, and the primary motor regions sequentially became dominant in controlling the output of the models. In addition, from the sensitivity analysis, a model can improve generalization
performance by only using more relevant neurons [San03b]. However, this approach relies on determining a suitable model, because they explicitly use the learned model to infer the dependencies.

There have been other approaches that do not depend on the model. One popular method applied in BMIs is the cellular tuning analysis which statistically reveals the firing modulation of each cell (or neuron) for specific movement parameters [Geo83]. A neuronal tuning curve, which is estimated by the probability distribution of firing rate of a given cell over entire hand position (or velocity) angles, is utilized to determine the tuning property of the cell. In this curve, if a cell firing shows higher probability in specific angle, we can see that the cell tunes its firing for a movement in that specific angle. Also, we can sort cells with the sharpness of their curves since shaper curves indicate finer tuning of cells associated with movement. However, this analysis does not take temporal aspects into account since it utilizes statistics over the entire data. Hence, it is difficult with this analysis to ascertain the dynamic properties of the neuronal relation with behavior.

In this chapter, we propose a model-independent approach to determine the spatio-temporal neuronal firing patterns by using of nonnegative matrix factorization (NMF) [Lee99, Lee01]. In its original applications, NMF was mainly used to provide an alternative method for determining sparse representations of images to improve recognition performance [Gui01 and Lee99]. D’Avella and Tresch have also proposed an extension of NMF to extract time-varying muscle synergies for the analysis of behavior patterns of a frog [dAv02]. The non-negativity constraints in NMF result in the unsupervised selection of sparse bases that can be linearly combined (encoded) to
reconstruct the original data. Our hypothesis is that NMF can similarly yield sparse bases for analyzing neural firing activity, because of the intrinsic non-negativity of the bin counts and the sparseness of spike trains.

We apply NMF to extract local features of neural bin counts in the same way sparse bases were obtained to describe the local features of face images. The basis vectors provided by NMF and their temporal encoding patterns are examined to determine how the activities of specific neurons localize to each segment of the reaching trajectory. We will show that the results from this model-independent analysis of the neuronal activity are consistent with the previous observations from the model-based analysis. And, the mixture of linear experts based on NMF bases and encodings will be designed to demonstrate how we can utilize NMF for improving models in BMIs.

**Nonnegative Matrix Factorization**

NMF is a procedure to decompose a non-negative data matrix into the product of two non-negative matrices: bases and encoding coefficients. The non-negativity constraint leads to a “parts-based” representation, since only additive, not subtractive, combinations of the bases are allowed. An $M \times N$ non-negative data matrix $X$, where each column is a sample vector, can be approximated by NMF as

$$X = WH + E$$

(8-1)

where $E$ is the error and $W$ and $H$ have dimensions $M \times r$ and $r \times N$, respectively. $W$ consists of a set of $r$ basis vectors, while each column of $H$ contains the encoding coefficients for every basis for the corresponding sample. The number of bases is selected to satisfy $r(M+N) < MN$ so that the number of equations exceed that of the unknowns.

This factorization can be described in terms of columns as
\( x_j \approx Wh_j, \) for \( j = 1, \ldots, N \) \hspace{1cm} (8-2)

where, \( x_j \) is the \( j \)th column of \( X \) and \( h_j \) is the \( j \)th column of \( H \). Thus, each sample vector is a linear combination of basis vectors in \( W \) weighted by \( h_j \). The non-negative constraints on \( W \) and \( H \) allow only additive combination of basis vectors to approximate \( x_j \). This constraint makes basis vectors to be visualized in the similar manner as the original sample. This is contrary to factorization by PCA, where the negative elements in basis vectors are allowed.

The decomposition of \( X \) into \( W \) and \( H \) can be determined by optimizing an error function between the original data matrix and the decomposition. Two possible cost functions used in the literature are the Frobenius norm of the error matrix \( \|X - WH\|_F^2 \) and the Kullback-Leibler divergence \( D_{KL}(X||WH) \). The non-negativity constraint can be satisfied by using multiplicative update rules discussed in Lee and Seung [Lee01] to minimize these cost functions. In this paper, we will employ the Frobenius norm measure, for which the multiplicative update rules that converge to a local minimum are given by

\[
H_{ij}^{(k+1)} = H_{ij}^{(k)} \frac{(W^TX)_{ij}}{(W^TWH)_{ij}} \\
W_{ij}^{(k+1)} = W_{ij}^{(k)} \frac{(XH^T)_{ij}}{(WHH^T)_{ij}}
\]

(8-3)

\( A_{ab} \) denotes element of a matrix \( A \) at \( a \)th row and \( b \)th column. Notice that the updates are based on the product of the current factor and a measure of quality of the current approximation [Wil03]. It has been proven in Lee and Seung [Lee01] that the Frobenius norm cost function is non-increasing under this update rule.
Factorization of Neuronal Bin Count Matrix

We will now apply the multiplicative update rule in (8-3) to the neuronal bin count matrix. The goal is to determine non-negative sparse bases for the neural activity, from which we wish to deduce the local spatial structure of the neuronal firings. These bases also point out common population firing patterns corresponding to the specific behavior. In addition, the resulting factorization yields a temporal encoding matrix that indicates how the instantaneous neural activity is optimally constructed from these localized representations. Since we are interested in the relationship between the neural activity and behavior, we would like to study the coupling between this temporal encoding pattern with the reaching movement of the primate, as well as the significance of the specific bases vectors, which represent neural populations.

Data Preparation

NMF is applied to two datasets; 3D food reaching data collected from an owl monkey (Belle), and 2D target reaching data collected from a Rhesus monkey (Aurora). For each dataset, the neuronal bin count matrix is formed as described in the followings.

3D food reaching data

In this recording session of approximately 20 mins (12000 bins), 104 neurons can be discriminated and there are 71 reaching movements for Belle. These reaching movements consist of three natural segments shown in Fig. 8-1. Based on the analysis of Wessberg et al. [Wes00], the instantaneous movement is correlated with the current and the past neural data up to 1 second (10 bins). Therefore, for each time instant, we form a bin count vector by concatenating 10 bins of firing counts (which correspond to 10-tap delay line in a linear filter) from every neuron.
Hence, if $x_j(n)$ represents the $n$th bin of neuron $j$, where $n \in \{1, \ldots, 12000\}$, a bin count vector at time instance $n$ is represented by $x(n) = [x_1(n), x_1(n-1), \ldots, x_1(n-9), x_2(n), \ldots, x_M(n-9)]^T$, where $M$ is the number of neurons. Since we are interested in determining repeated spatio-temporal firing patterns during the reaching movements, only the bin counts from time instances where the primate’s arm is moving are considered. There is a possibility that in the selected training set some neurons never fire (this reduces $M$ from 104 to 99). The rows corresponding to these neurons must be removed from the bin count matrix, since they tend to cause instability in the NMF algorithm. In addition, to prevent the error criterion from focusing too much on neurons that simply fire frequently (although the temporal structure of their activity might not be significant for the task), the bin counts in each row (i.e., for each neuron) of the data matrix are normalized to have the unit length in its two norm. In general, if $M$ neurons are considered for a total of $N$ time instances, the data matrix $X$ has dimension $(10M) \times N$. Since the entries of the data
matrix are bin counts, they are guaranteed to be non-negative. Accounting for 71 movements, there are \( N = 2143 \) time instances for this data.

**2D target reaching data**

In this BMI, the primate continuously moves the arm to track a target on screen. Since there is no pause between movements as is the case in food reaching, a continuous neuronal bin count data collected from certain segment in a recording session is used for NMF. The length of the data is chosen to be 500 seconds (5,000 samples).

Accounting for temporal firing patterns, we could embed the bin count data for each neuronal channel. However, preliminary experimental studies have revealed that such an embedded data matrix cannot be factorized to yield sparse and local representations of neuronal firing activities. Hence, we utilize the multiresolution representation of neuronal firings [Kim05c] as discussed in the previous chapter instead of time delay embedding. The bin widths used here are determined to be 160ms, 320ms, and 640ms since including shorter bin widths only increases complexity of factorization without providing useful NMF bases.

With three bin widths and 185 neurons, the bin count matrix set up for NMF has a dimension \( 555 \times 5000 \). The hand position trajectory is composed of x-, and y- coordinates in 2D space. Each row in the bin count matrix is normalized as above.

**Analysis of Factorization Process**

In the application of NMF to a given neural firing matrix, there are a few important issues that must be addressed: the selection of the number of bases, the local minima of the NMF algorithm, and understanding how NMF can find repeating patterns.
Choice of the number of bases

The problem of the choice of the number of bases can be addressed in the framework of model selection. A number of model selection techniques (e.g. the cross-validation) can be utilized for finding the optimal number of bases. In this dissertation, we choose to adopt a selection criterion that has been recently developed for clustering. The criterion is called the index $I$, which has been used to indicate the cluster validity [Mal02]. This index has shown consistent performance of selecting the true number of clusters for various experimental settings. The index $I$ is composed of three factors as,

$$I(r) = \left( \frac{1}{r} \cdot E_r \cdot D_r \right)^p$$

(8-4)

where $E_r$ is the approximation error (Frobenius norm) for $r$ bases, and $D_r$ is the maximum Euclidean distance between bases such that,

$$D_r = \max_{i,j=1}^r \| w_i - w_j \|.$$  

(8-5)

The optimal $r$ is the one that maximizes $I(r)$. We will utilize this index to determine the optimal $r$ for NMF with $p=1$.

How does NMF find repeated patterns?

Although a rigorous proof that the NMF bases will discover repetitive patterns in the neuronal data is not provided here, by intuition we can argue that NMF will prefer to minimize the cost function by selecting repetitive firing patterns in the bases. The reason for this is the fact that, if a repeated firing sequence is not in the space spanned by the linear combination of the selected bases, the error contribution of this pattern will scale up by the number of occurrences of the pattern. On the other hand, if a firing sequence
occurs only a few times, then the cost of not representing this sequence in the bases will be relatively much smaller.

Donoho and Stodden have shown that a unique solution of NMF is possible under certain conditions [Don04]. They have shown through a geometrical interpretation of NMF that if the data are not strictly positive, there can be only one set of non-negative bases which spans the data in the positive orthant. With an articulated set of images obeying three rules (a generative model, linear independence of generators, and factorial sampling), they showed NMF identifies the generators, or “parts” of images. If we consider our neuronal bin-count matrix, each row contains many zero entries (zero bin counts) even after removing non-firing neurons since most neurons do not fire continuously once in every 100ms window during the entire training set. Therefore, our neuronal data are not strictly positive. This implies that the existence of a unique set of non-negative bases for the neuronal bin-count matrix is warranted. The question still remains if the NMF basis vectors can find the generative firing patterns for the neural population by meeting the three conditions mentioned above. Here, we discuss the neuronal bin-count data with respect to these conditions.

As stated previously, we have demonstrated through sensitivity analysis that the specific neuronal subsets from the PP, PMd, and M1 regions were sequentially involved in deriving the output of the predictive models during reaching movements [San03a]. Hence, the bin-count data for the reaching movement will contain increasing firing activity of the specific neuronal subset on local partitions of the trajectory. Due to binning, it is possible that more than one firing patterns is associated with a single data sample. This analysis leads to a generative model for the binned data in which data
samples are generated by linear combination of the specific firing patterns with non-negative coefficients. Also, these firing patterns will be linearly independent since the neuronal subset in each firing patterns tends to modulate firing rates only for the local part of trajectory. The third condition of factorial sampling can be approximately satisfied by the repetition of movements in which the variability of a particular firing pattern is observed during the entire data set. However, a more rigorous analysis is necessary to support the argument that the set of firing patterns is complete in factorial terms. Therefore, we expect that the NMF solutions may be slightly variable reflecting the ambiguity in the completeness of factorial sampling. This might be overcome by collecting more data for reaching movements, and will be pursued in future studies.

A simple insight into the NMF algorithm may also help us understand how it captures the repetitive patterns. The update equation (8-3) is restated here for convenience as

\[
H_{aj}(k + 1) = H_{aj}(k) \cdot \frac{(W^T X)_{aj}}{(W^T WH)_{aj}}
\]

\[
W_{ia}(k + 1) = W_{ia}(k) \cdot \frac{(XH^T)_{ia}}{(WHH^T)_{ia}}
\]

We have mentioned that the new update is the product of the current factor and the current quality of the approximation. We can see that this quality of the approximation is estimated by the inner product. Let us first look at the update of each column of \( W \).

\((XH^T)_{aj}\) and \((WHH^T)_{aj}\) can be rewritten in the vector form as

\[
(XH^T)_{ia} = x_i^T h_a
\]

\[
(WHH^T)_{ia} = \hat{x}_i^T h_a
\]
where \( \mathbf{x}_i \) denotes the \( i^{th} \) row vector of \( \mathbf{X} \), and \( \mathbf{h}_a \) denotes the \( a^{th} \) row vector of \( \mathbf{H} \). \( \hat{\mathbf{x}}_i \) is also the \( i^{th} \) row vector of \( \hat{\mathbf{X}} \) where \( \hat{\mathbf{X}} = \mathbf{W}\mathbf{H} \). Since we normalize each row of \( \mathbf{X} \), \( \mathbf{x}_i^T \mathbf{h}_a \) depends on the angle between two vectors. So, it reflects the correlation between \( \mathbf{h}_a \) and the temporal pattern of each neuron (or delayed version of it). Then, some of the entries in the \( a^{th} \) basis with similar temporal patterns as \( \mathbf{h}_a \) are updated with relatively larger (positive) amounts. This is analogous to the spherical \( k \)-means clustering [Dhi01] in which clusters are organized based on the metric of the cosine similarity, in the sense that the channels with similar temporal patterns are clustered with \( \mathbf{h}_a \) as center, being encoded in the \( a^{th} \) basis. This analogy is also supported by the fact that seeding NMF using spherical \( k \)-means clustering improves convergence speed [Wil03].

After updating the \( a^{th} \) basis vector, \( \mathbf{h}_a \) is updated in a similar manner. The update equation can be rewritten as,

\[
(W^T X)_{ia} = w_a^T x_{i.} \\
(W^T WH)_{ia} = w_a^T \hat{x}_j
\]

where \( w_a \) denotes the \( a^{th} \) column of \( \mathbf{W} \) or the \( a^{th} \) basis vector, and \( x_{i.} \) is the \( i^{th} \) column of \( \mathbf{X} \) or the \( i^{th} \) sample vector. The \( i^{th} \) element of \( \mathbf{h}_a \) is then a correlation measure of \( w_a \) with the \( i^{th} \) sample vector. Note that the numerator of the update for \( \mathbf{h}_a \), i.e., \( w_a^T \mathbf{X} \), can be regarded as a weighted sum of rows of \( \mathbf{X} \) where more weights are imposed on the rows with similar patterns to \( \mathbf{h}_a \). This update is very similar to the numerator in the update rule of the spherical \( k \)-means clustering that is given by

\[
c_j(k + 1) = \frac{\sum_{i \in \pi_j} x_i}{\sum_{i \in \pi_j}}, \quad \pi_j = \left\{ x : x^T c_j(k) \geq x^T c_m(k), m \neq j \right\}
\]
for the $j^{th}$ cluster center [Dhi01]. With this update, $h_a$ will encode the information of time when the spatial pattern of the data similar to $w_a$ occurs.

In short, if there is a repeating pattern in a given sparse neural activity dataset, which implies that a subset of neurons have a common temporal pattern, this subset and its common temporal pattern would be encoded in the NMF bases and encodings, respectively.

**Local minima problem**

We can utilize these insights to reduce the effect of the local minima. For instance, we can initialize $H$ with positive random numbers and $W$ with a fixed positive constant. Then the quantity of $\hat{x}_i^T h_a = (WHH^T)_{ia}$ becomes the same for every row, since every row of $W$ has the same entries and it is multiplied by the $a^{th}$ column of the matrix $HH^T$. Therefore, the update of each entry in the bases can be solely dependent upon the angle between $x_i$ and $h_a$. Then the final solution will be mainly dependent upon the initialization of $H$ and have similar sensitivity to initial condition as spherical k-means algorithm does.

**Case Studies A: 3D Food Reaching**

The NMF algorithm is applied to the described neuronal data matrix prepared using ten taps, $M = 91$ neurons. The NMF algorithm with 100 independent runs results in $r = 5$ bases for which the index $I$ is maximized. The means and the standard deviations of the normalized cost (Frobenius norm of error between approximation and the given data matrix divided by the Frobenius norm of the data only) for 100 runs are $0.8399 \pm 0.001$. This implies that the algorithm approximately converges to the same solution with different initial conditions. Note that we initialize $H$ as stated in the previous section.
In Fig. 8-2, we show the resulting basis vectors (columns of $W$) for the bin counts (presented in matrix form where columns are different neurons and rows are different delays), as well as their corresponding time-varying encoding coefficients (rows of $H$) superimposed on the reaching trajectory coordinates of three consecutive movements. Using these time-synchronized neural activity and hand trajectory recordings, it is also possible to discover relationships between firing patterns and certain aspects of the movement. Since NMF is looking for an optimal linear approximation of the data with few bases (which can be realized by discovering latent structures in the data [Lee01]), an efficient representation of the complete firing activity can be achieved by selecting the bases such that each basis represents a repeated spatio-temporal firing sequence. For

Figure 8-2. The NMF results for food reaching; Left) the five bases. Right) their corresponding encoding signals (thick solid line) overlaid on the 3D coordinates of the hand trajectory (dotted lines) for three consecutive representative reaching tasks (separated by the dashed lines).
example, from the basis vectors in the left panel of Fig. 8-2, we observe that firings of the neurons in group-\(b\) are followed by firings of the neurons in group-\(a\) (the bright activity denoted by \(b\) occurs earlier in time than the activity denoted by \(a\), since increasing values in the vertical axis of each basis indicates going further back in time). Thus, NMF effectively determines and summarizes this sparse firing pattern that involves a group of neurons firing sequentially. Their relative average activity is also indicated by the relative magnitudes of the entries of this particular basis.

We can assess the repeatability of a certain firing pattern summarized by a basis vector by observing the time-varying activity of the corresponding encoding signal (the corresponding row of \(H\)) in time. An increase in this coefficient corresponds to a larger emphasis to that basis in reconstructing the original neural activity data. In the right panel of Fig. 8-2, we observe that all bases are activated regularly in time by their corresponding encoding signals (at different time instances and at different amplitudes). For example, the first basis is periodically activated to the same amplitude, whereas the activation amplitude of the third basis varies in every movement, which might indicate a change in the role of the corresponding neuronal firing pattern in executing that particular movement. The periodic activation of encodings also indicates the bursting nature of the spatio-temporal repetitive patterns. Hence, the NMF bases tend to encode synchronous and bursting spatio-temporal patterns of neural firing activity.

From the NMF decomposition, we observe certain associations between the activities of neurons from different cortical regions and different segments of the reaching trajectory. In particular, an analysis based on Fig. 8-2 indicates that neurons in
PP and M1 repeat similar firing patterns during the reach from rest to food. This assessment is based on the observation that bases three, four and five, which involve firing activities from neurons in these regions, are repeatedly activated by the increased amplitude of their respective encoding coefficients. Similarly, neurons in M1 are repeatedly activated during the reach to and from the mouth (bases one and two). These observations are consistent with the sensitivity analysis that was conducted through trained input-output models (such as the Wiener filter and RMLP) [San03a]. Table 8-1 compares the neurons, which were observed to have the highest sensitivity from trained models, and the neurons that have the largest magnitudes in each NMF basis. We can see that neurons from NMF are a subset of neurons obtained from the sensitivity analysis. It is also worth stating that NMF basis provides more information than the model-based sensitivity analysis since it determines the synchronous spatio-temporal patterns while the sensitivity analysis only determines individual important neurons. Finally, we would like to reiterate that the analysis presented here is solely based on the data.

<table>
<thead>
<tr>
<th>Regions</th>
<th>PP</th>
<th>M1</th>
<th>PMd</th>
<th>M1-ipsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>The high sensitive neurons</td>
<td>4,5,7,22,26,</td>
<td>38,45</td>
<td></td>
<td>93,94</td>
</tr>
<tr>
<td>through RMLP</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The largest-magnitude neurons</td>
<td>7,23, 29</td>
<td>45</td>
<td></td>
<td>93,94</td>
</tr>
<tr>
<td>in NMF bases</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Case Study B: 2D Target Reaching**

The NMF algorithm is applied to the neuronal bin count matrix generated by using of multiresolution analysis with three scales and $M = 185$ neurons. With 100 independent runs, the index I is again maximized for $r = 5$ bases. The means and the standard deviations of the normalized cost for 100 runs turn out to be $0.425 \pm 0.001$, which again reveals robustness of the NMF algorithm about initial conditions.
The resulting basis vectors and the corresponding encoding coefficients (row of H) along with the hand position trajectories are demonstrated in Fig. 8-3. Unlike previous observations in food reaching, the repetition of a firing pattern is not clearly shown in time-varying encodings due to the irregular nature of movement. Also, the basis vectors are not as sparse as the ones for food reaching.

Despite these rather complicated results of basis and encoding, we can extract some information about firing patterns encoded in each basis vector by looking into its counterpart encoding time series. Let us first segment the hand trajectory into three sample regions; divided by dotted lines in the right panel of Fig. 8-3. Although the trajectories in each region are different, they have similar pattern; increase in both x, and

![Figure 8-3. The NMF results for target reaching; Left) the five bases. Right) their corresponding encoding signals (thick black line) overlaid on the 2D coordinates of the target reaching trajectory (x: blue, y: red).](image-url)
y directions – decrease in both directions – increase again – and decrease at the end. For each region, encodings of individual basis vectors exhibit characteristic patterns. The first encoding tends to increase its magnitude around points when hand trajectory starts to decrease from the positive peaks. The second encoding tends to increase when hand trajectory starts to move in positive direction. The third encoding increases for moving of hand in negative direction. The fourth encoding exhibits peaks in the middle of moving in positive direction, and the fifth encoding exhibits peaks in negative direction.

Also, the temporal sequence of contributions of neurons from each cortical area can be approximated from this figure. In each sample region, there seems to be a sequence of dominance of each basis vector by observation of encoding patterns; the individual encodings tend to increase following a particular sequence such as 3-2-5-4-1 (indices of encodings from top to bottom in Fig. 8-3. Since the movement is continuous, this sequence can be made circular (e.g. 5-4-1-3-2). This observation of sequence may be linked with neurophysiologic functions of each cortical area, which will be an interesting future research topic.

Based on these observations, we statistically analyze each encoding associated with movement; the hand position samples when each encoding series moves around the peaks are collected, and the average position among these is estimated. This average position indicates the relative location of hand positions when the neuronal firing patterns determined in each basis vector appears. Figure 8-4 shows the average hand positions for the five basis vectors in 2D space. Note that the estimated standard deviation for each collected samples is so large that there are wide overlaps between distributions of
collections. But this distribution approximately explains which part of the trajectory each neuronal firing pattern is related with.

It is noteworthy that the distributions of hand positions for some NMF bases are similar, typically located around 45° or 200° angles. This is due to the fact that the probability that hand position is located around those angles is relatively higher. This fact is empirically illustrated in Fig. 8-5 where the probability that the hand is positioned in each of 16 angle bins. These bins are obtained by partitioning the 2D polar axis into equally spaced 16 angle bins. From this figure, we can see that the hand is located at around 45° or 200° with higher probability.

Figure 8-4. The hand position samples collected along with peaks in each NMF encoding (left), and the mean and variance of each set (right). The number marked with each dot denotes the corresponding basis in Fig. 8-3 (number in order of top to bottom).
Another analysis of encodings can be conducted based on cellular tuning [Geo83]. In the cellular tuning analysis, each neuronal modulation with respect to the angle of hand position is investigated. Similarly, tuning property of neuronal pattern in each NMF basis is investigated through its counterpart encodings. We first partition the angle in 2D space into 16 angle bins from \( -\pi \) to \( \pi \) in radian as described above. Then, a \( 16 \times N \) Boolean matrix \( A \) is created of which each column vector \( a_n \) consists of zero elements except only one unit value; \( a_n = [0, \ldots, 0, 1, 0, \ldots]^{T} \). The location of 1 in each vector points out the angle bin where the current angle of hand position at time instance \( n \) belongs to. Since the encoding matrix \( H \) has all nonnegative elements and the large value of encoding indicates activation of the corresponding neuronal pattern, each row of \( HA^{T} \) indicates tuning property of the corresponding basis estimated over entire data samples. Figure 8-6 presents each row of \( HA^{T} \) that shows a tuning curve of individual neuronal patterns encoded in each basis.
It is interesting to compare Fig. 8-6 with Fig. 8-4. We can see that the angel of each average hand position in Fig. 8-4 matches tuning curves in Fig. 8-6. For instance, the first average hand position has an angle around $\pi/3$ at which the first tuning curve exhibits a peak. Other hand positions and the corresponding tuning curves also match each other. Hence, we can conclude that NMF basis vectors and encodings represent tuning properties of neuronal firing patterns appearing in basis. It also demonstrates that the neurons clustered by each basis have the common characteristics associated with behavior. We compare the neurons with large weights in basis vectors of NMF with important neurons selected by the sensitivity analysis, summarized in table 8-2. The neurons are selected empirically by looking into elements in NMF basis vectors corresponding to each neuron. The selection by NMF basis is significantly compatible with that from the sensitivity analysis.

Figure 8-6. Tuning curve of neuronal firing patterns encoded in each NMF basis for 16 angle bins.
Table 8-2. Comparison of important neurons: target reaching.

<table>
<thead>
<tr>
<th>Regions</th>
<th>PMd</th>
<th>M1</th>
<th>S1</th>
<th>SMA</th>
<th>M1_ipsi</th>
</tr>
</thead>
<tbody>
<tr>
<td>The high sensitive neurons</td>
<td>15, 54</td>
<td>68,69,73,76,78,80,81,84,92,96,99,101,104,107,108,110</td>
<td>149</td>
<td>167</td>
<td></td>
</tr>
<tr>
<td>through RMLP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The largest-magnitude neurons</td>
<td>7,41,45,48</td>
<td>67,73,76,78,80,81,84,89,92,99,104,108,110,114,121</td>
<td>145, 167</td>
<td>169, 167</td>
<td></td>
</tr>
<tr>
<td>in NMF bases</td>
<td>54,66,</td>
<td></td>
<td>149</td>
<td></td>
<td>177</td>
</tr>
</tbody>
</table>

Model Improvement Using NMF

We will demonstrate a case study of performance improvement in predicting hand positions in a 3D food reaching BMI by utilizing NMF. We will compare the performance of two systems; the Wiener filter directly applied to the original binned data and the mixture of multiple linear filters based on the NMF bases and encodings.

The straight Wiener filter is directly applied to the neural firing data to estimate the three coordinates of the primate’s hand position. The properties and modeling of the Wiener filter are discussed in chapter 3.

The mixture of multiple models employs the NMF encodings as mixing coefficients. A NMF basis is used as a window function for the corresponding local model. Therefore, each model sees a given input vector through a different window and uses the windowed input vector to produce the output. Then the NMF encodings are used to combine each model’s output to produce the final estimate of the desired hand position vector. This can be described in the following equation,

\[ \hat{d}_c(n) = \sum_{k=1}^{K} h_k(n) (z_k(n)^T g_{k,c} + b_{k,c}), \]  

(8-10)

where \( h_k(n) \) is an NMF encoding coefficient for the \( k^{th} \) basis at \( n^{th} \) column (i.e., time index), \( g_{k,c} \) is the weight vector of the \( k^{th} \) model for the \( c^{th} \) coordinate \( (c \in \{x,y,z\}) \), and
$b_{k,c}$ is the y-intercept of the $k^{th}$ model for the $c^{th}$ coordinate. $z_k(n)$ is the input vector windowed by the $k^{th}$ NMF basis. Its $i^{th}$ element is given by

$$z_{k,i}(n) = x_i(n) \cdot w_{k,i}$$

(8-11)

Here, $x_i(n)$ is the normalized firing count of the neuron $i$ at time instance $n$, and $w_{k,i}$ is the $i^{th}$ element of the $k^{th}$ NMF basis. $g_{k,c}$ and $b_{k,c}$ can be estimated based on the MSE criterion by using of the stochastic gradient algorithm such as the normalized least mean square (NLMS). The weight update rule of the NLMS for each model is then given by

$$g_{k,c}(n + 1) = g_{k,c}(n) + \eta \frac{1}{\beta + \|z_k(n)\|^2} h_k(n)e_c(n)z_k(n)$$

$$b_{k,c}(n + 1) = b_{k,c}(n) + \eta \frac{1}{\beta + \|z_k(n)\|^2} h_k(n)e_c(n)$$

(8-12)

where $\eta$ is the learning rate and $\beta$ is the normalization factor. $e_c(n)$ is the error between the $c^{th}$ coordinate of the desired response and the model output.

In the experiment, we divided the data samples into 1771 training samples and 372 test samples for 3D food reaching dataset. The parameters are set as $\{\eta, \beta, K\} = \{0.01, 1, 5\}$. The entire training data set is presented 60 times for the weights to converge. The performance of the model is evaluated on the test set by two measures; the correlation coefficient (CC) between desired hand trajectory and the model output trajectory, and the mean squared error (MSE) normalized by the variance of the desired response. Table 8-3 presents the evaluation of the performance of two systems. It shows a significant improvement in performance with the mixture of models based on NMF factorization.

<table>
<thead>
<tr>
<th></th>
<th>CC(x)</th>
<th>CC(y)</th>
<th>CC(z)</th>
<th>MSE(x)</th>
<th>MSE(y)</th>
<th>MSE(z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wiener filter</td>
<td>0.5772</td>
<td>0.6712</td>
<td>0.7574</td>
<td>0.4855</td>
<td>0.3468</td>
<td>0.2460</td>
</tr>
<tr>
<td>NMF mixture</td>
<td>0.7147</td>
<td>0.7078</td>
<td>0.8076</td>
<td>0.2711</td>
<td>0.2786</td>
<td>0.1627</td>
</tr>
</tbody>
</table>
To quantify the performance difference between the Wiener filter and the mixture of multiple models, we can apply a statistical test based on the mean squared error (MSE) performance metric as proposed in Kim et al. [Kim05a]. The details of this statistical test procedure will be revisited later in chapter 8. We perform the t-test on both modeling outputs with a significance level of 0.01 or 0.05. The null hypothesis is rejected with both significance levels, resulting in the \( p \)-values of 0.0023. Therefore, the statistical test of the performance difference demonstrates that the mixture of multiple models based on NMF improves the performance significantly compared to the standard Wiener filter.

In this case study, we show that the bases and encodings of NMF can be effectively used for modeling the transfer function from neuronal firing patterns to hand movements. Although we demonstrate a simple mixture of linear experts here, there are still a lot of possibilities to enhance performance by investigation of NMF results for BMIs modeling. Future works will pursue this interesting subject.

**Discussions**

The results presented in the previous case study are a representative example of a broader set of NMF experiments performed on this recording. Selection of the number of taps and the number of bases \( (r) \) is dependent on the particular stimulus or behavior associated with the neural data. Although we have used a model selection method originally developed for clustering, and did not provide full justification that this index is suitable to NMF, the main motivation is to demonstrate that the problem of selecting the number of bases can be addressed in the context of model selection. This will be pursued in future research.

As we discussed in above, NMF attempts to detect repeating firing patterns and assigns a basis vector for each pattern, simply because it is the optimal strategy to
minimize the error cost function. Consequently, the number of patterns that can be distinctly represented by NMF is limited by the number of bases. A very small number of bases will lead to the combination of multiple patterns into a single non-sparse basis vector. At the other extreme, a very large number of bases will result in the splitting of a pattern into two or more bases, which have similar encoding coefficient signals in time. In these situations, the bases under consideration can be combined into one basis.

It is noteworthy that the performance measures of the Wiener filter designed in this chapter are not identical to those of the Wiener filter in table 6-1 since the training and test sets are different. For the Wiener filter in table 6-1, 10,000 training samples and 3,000 test samples are used, whereas for the one in this chapter, only 1,771 samples for training and 372 samples for test are used. Hence, the poorer generalization performance of the Wiener filter in this chapter is due to a smaller number of training samples.

It is intriguing that the mixture of models based on NMF generalizes better than the Wiener filter despite the fact that the mixture contains much more model parameters. However, each model in the mixture receives the inputs processed by the sparse basis vector. Therefore, each model learns the mapping between only a particular subset of neurons and hand trajectories, and the effective number of parameters for each model is much less than the total number of input variables. Moreover, further overfitting is avoided by combining the outputs of local models by the sparse encodings of NMF.
CHAPTER 9
REAL TIME NEURONAL SUBSET SELECTION

The analytical methods including the sensitivity analysis, the cellular tuning analysis, and NMF extract neuronal population properties use the entire data by assuming stationarity. Hence, the information given by these methods may not be sufficient for the analysis of neuronal population functions that may be nonstationary. These facts lead us to consider a new analytical tool for finding the nonstationary properties, in both time and the space of the electrodes, of neuronal population related with behavior. This is evident in the observation that only certain subset of neurons is involved in a particular movement, and the composition of subset varies over time [Kim05b].

In order to develop an analytical solution, finding the nonstationary neuronal relationships with behavior is recognized as the problem of tracking time-variant MIMO system in which only a subset of input channels contributes to desired response at a given moment. Thus, we begin our development in the basis of current tracking methods for nonstationary systems. Similar to the sensitivity analysis, the basic idea is to extract the real time information of the neuronal relationships to the model outputs through time-varying model parameters. There have been a variety of adaptive methods to adjust parameters for tracking time-variant systems including least mean squares (LMS), and recursive least squares (RLS) [Hay96a]. Those adaptive algorithms, however, may not be suitable to exploit the spatial structure in the multi-channel data (as in multiple neuronal channels) since their tracking capabilities are basically guaranteed for single input time series. Also, the constant control parameters in LMS (e.g. a step size), or in RLS (e.g. a
forgetting factor) prevent the algorithms from tracking the nonstationary system more accurately (although we can vary such parameters in time, it is a nontrivial problem).

In order to overcome the limitations of current tracking methods, we propose a new adaptive system modeling which can exploit the spatial structure of neuronal population in real time by augmenting a spatial filter. In this structure, the outputs from filters in individual channels are filtered again using an on-line variable selection algorithm [Kim04]. This selection algorithm enables us to find a subset of neurons relevant to a particular movement in every time instance. Hence, we can extract the information of which neuronal subset is correlated with behavior at a particular moment. We believe that this method provides an analytical tool for extracting the nonstationary properties of the relationship between neuronal population and the associated behavior.

In the design of the real time neuronal subset selection algorithm, there are several issues to be addressed. Firstly, the movement is correlated with the temporal pattern of neuronal activity. Secondly, the neuronal firing data are nonstationary in time and in space (over neuronal channel). Finally, only a subset of neurons may be involved in a particular trajectory of movement.

In our proposed algorithm, the correlation of each neuron with behavior is measured at the filtered output of each input channel, which can incorporate the temporal firing patterns of individual neurons. The filter parameters in every neuronal channel are adjusted in real time by LMS to track the nonstationarity of neuronal contributions to movement. In order to enhance the LMS algorithm with a selective mechanism sensitive to the spatial multi-channel structure, we include a second stage spatial filter to the filtered channel outputs. This spatial filter imposes time varying weighting on each
channel to track in space the time-varying spatial structure. The spatial filter parameters are adapted in real time by using an on-line variable selection algorithm. By virtue of the selection scheme in this algorithm, the spatial filter can be sparse and select a subset of neuronal channels at every time instance.

An on-line variable selection algorithm based on LAR has been developed to select a subset of input variables relevant to desired response in a sparse linear time-variant system. If we constrain the $L_1$-norm of coefficients such that the LAR procedure stops at certain number of steps less than the total number of variables, then the regression model only has nonzero coefficients in a subset of the input variables. However, as introduced in chapter 4, LAR processes the entire data to adjust coefficients based on a stationary assumption. In order to devise an on-line algorithm of the LAR procedure, we utilize the RLS type recursion for the input covariance matrix and the cross-correlation vector between input and desired response, and modify the LAR algorithm accordingly. By this modified LAR algorithm implemented in real time, a subset of channels can be selected based on correlation with desired response at every time instance.

We will demonstrate real time neuronal subset selection in BMIs for two datasets; 3D food reaching data of Belle, and 2D target reaching data of Aurora. The experimental results will show the nonstationary characteristics of the contributions of individual neurons to movements.

The chapter starts by introducing the on-line variable selection by modifying the LAR algorithm. Next, the architecture and procedure of real time neuronal subset selection method will be presented, followed by the discussion about determination of
On-Line Variable Selection

Correlation between inputs and the desired response can be accomplished by recursively updating the correlation vector. The input covariance matrix can also be estimated recursively. If one decouples the variable selection part from the model update part in LAR, we can select the input variables locally with recursive estimates of correlations. The modified version of LARS for on-line variable selection is described as follows. We rewrite table 4-1 here for convenience in table 9-1.

Table 9-1. Procedure of the LAR algorithm: revisited.

Given an $N \times M$ input matrix $X$ (each row being $M$-dimensional sample vector), and an $N \times 1$ desired response matrix $Y$, initialize the model coefficient $\beta_i = 0$, for $i = 1,\ldots, M$, and let $\beta = [\beta_1, \ldots, \beta_M]^T$.

Then the initial LAR estimate becomes, $\hat{Y} = X\beta = 0$.

Transform $X$ and $Y$ such that

$$
\begin{align*}
\frac{1}{N} \sum_{i=1}^{N} x_{ij} = 0, \\
\frac{1}{N} \sum_{i=1}^{N} y_i^2 = 1, \\
\frac{1}{N} \sum_{i=1}^{N} y_i = 0 \quad \text{for} \quad j = 1,\ldots, M.
\end{align*}
$$

(a). Compute the current correlation $c = X^T (Y - \hat{Y})$.

(b). Find $C_{\max} = \max_j |c_j|$, and a set $A = \{j: |c_j| = C_{\max}\}$.

(c). Let $X_a = \{\ldots, \text{sign}(c_j)x_j, \ldots\}$ for $j \in A$.

(d). Let $\Phi = X_a^T X_a$, and $\alpha = (1_a \Phi^{-1} 1_a)^{-1}$, where $1_a$ is a vector of one’s with a length equal to size of $A$.

(e). Compute the equiangular vector $\mu = X_a (\alpha \Phi^{-1} 1_a)$ that has the unit length. Note that $X_a \mu = \alpha 1_a$ (angles between all inputs in $A$ and $\mu$ are equal).

(f). Compute the step size, $\gamma = \min^+_{j \in A^c} \left\{ \frac{C_{\max} - c_j}{\alpha - \theta_j}, \frac{C_{\max} + c_j}{\alpha + \theta_j} \right\}$

where $\min^+$ indicates considering only positive minimum values over possible $j$.

(g). Compute $\theta_j$ which is defined as the inner product between all inputs and $\mu$ such as, $\theta_j = X^T \mu$.

(h). Update $\hat{Y}_+ = \hat{Y} + \gamma \mu$.

Repeat (a)-(h) until all inputs join the active set $A$, or $\sum_j |\beta_j|$ exceeds the given threshold.
Let us first analyze the LAR procedure illustrated in table 9-1. As stated in Kim et al. [Kim04], the time correlation (table 9-1 (a)) at a given step for the \( k^{th} \) step of variable selection can be simply updated without computing residuals,

\[
c_j(k) = c_j(k) - \gamma \theta_j
\]  

(9-1)

Hence, the update procedure of table 9-1(h) can be removed. The initial correlation \( c_j(0) \), which represents the correlation between inputs and desired response can be estimated outside of the LAR routine. Instead of computing the correlation with entire data, we can recursively estimate the correlation using a forgetting factor, given by

\[
p(n) = \rho p(n-1) + d(n)x(n)
\]  

(9-2)

where \( \rho \) is the parameter controlling memory depth, and \( x(n) \) is an \( 1 \times M \) input vector at time instance \( n \). This estimate of the correlation vector, \( p(n) \) is utilized by the LAR routine such that \( c_j(0) = p_j(n) \).

For the computation of the covariance matrix \( \Phi \) in table 9-1(d), we also estimate the input covariance matrix using the leaky integrator in the same way as (9-2),

\[
R(n) = \rho R(n-1) + x(n)^T x(n)
\]  

(9-3)

where \( \delta \) is another step size for covariance estimation. This matrix is not directly used in table 9-1(d) since \( \Phi \) is the covariance of only subset of inputs. Also, the input vectors are multiplied by the sign of correlations before computing \( \Phi \). Therefore, we introduce a diagonal matrix \( S \) whose elements are signs of \( c_j(k) \) for \( j \in A \). Then, \( \Phi \) can be computed using \( R(n) \) and \( S \) as

\[
\Phi = SR_aS
\]  

(9-4)

where \( R_a \) is \( L_a \times L_a \) (\( L_a \) is the length of \( A \) in table 4-1) matrix representing covariance among the selected input variables. \( R_a \) can be given by elements of \( R(n) \), i.e., \( r_{ij} \) for \( i, j \in \)
To remove the computation of the equiangular vector \( \mu \) that requires a batch computation, we incorporate table 9-1(e) into table 9-1(g) such that

\[
\theta_j = X^T \mu = X^T X_a (\alpha \Phi^{-1} 1_a) = \alpha X^T X_a \Phi^{-1} 1_a
\]

(9-5)

However, \( X^T X_a \) is nothing but the \( j^{th} \) columns of \( R(n) \), for \( j \in A \) followed by multiplication with \( S \). So, if we define \( R_{\text{acol}} \) to be a submatrix of \( R(n) \) consisting of the \( j^{th} \) columns for \( j \in A \), then

\[
\theta_j = \alpha R_{\text{acol}} S \Phi^{-1} 1_a
\]

(9-6)

Hence, using \( \Phi \) obtained by \( R(n) \) and \( S \), we can compute \( \alpha \) and consecutively \( \theta_j \) for \( j \in A \). This modification removes the computation of the equiangular vector in table 4-1(e), which is not directly required for computing \( \theta_j \) and \( \gamma \) in (f). Table 9-2 summarizes this modified version of the LARS algorithm.

**Table 9-2. The modified LAR algorithm for on-line variable selection.**

Given an \( N \times M \) input matrix \( X \) (each row being \( M \)-dimensional sample vector), and an \( N \times 1 \) desired response matrix \( Y \), initialize \( p(0) = 0 \) and \( R(0) = 0 \).

Transform \( X \) and \( Y \) such that

\[
\frac{1}{N} \sum_{i=1}^{N} x_{ij} = 0, \quad \frac{1}{N} \sum_{i=1}^{N} x_{ij}^2 = 1, \quad \text{and} \quad \frac{1}{N} \sum_{i=1}^{N} y_i = 0 \quad \text{for} \quad j = 1, \ldots, M.
\]

Update the correlation: \( p(n) = \rho p(n-1) + d(n) x(n) \)

Update the input covariance: \( R(n) = \rho R(n-1) + x(n)^T x(n) \)

(a) \( c(0) = p(n) \).

(b) \( C_{\text{max}} = \max_j \left| c_j \right|, \) and \( A = \{ j : |c_j(k)| = C_{\text{max}} \} \).

(c) Compute a diagonal matrix \( S \) with elements of sign of \( c_j(k) \) for \( j \in A \).

(d) \( \Phi = SR_a S \),

where \( R_a \) is submatrix of \( R(n) \) with \( j^{th} \) rows and \( j^{th} \) columns for \( j \in A \).

(e) \( \alpha = (1_a^T \Phi^{-1} 1_a)^{-1} \)

(f) \( \theta_j = \alpha R_{\text{acol}} S \Phi^{-1} 1_a \),

where \( R_{\text{acol}} \) is a matrix consisting of \( j^{th} \) columns of \( R(n) \) for \( j \in A \).

(g) Compute the step size, \( \gamma = \min_{j \in A C} \frac{C_{\text{max}} - c_j}{\alpha - \theta_j} \cdot \frac{C_{\text{max}} + c_j}{\alpha + \theta_j} \).

(f) Update correlation: \( c_j(k) = c_j(k) - \gamma \theta_j \).

Repeat b – f until all inputs join the active set \( A \), or \( \sum_j \left| \beta_j \right| \) exceeds the given threshold.
On-Line Channel Selection Method

The on-line variable selection algorithm has been demonstrated to track linear time-variant system by combination with a linear adaptive system [Kim04]. It outperforms traditional algorithms including LMS when only subset of input variables are related with the target signals at each time period. However, this algorithm lacks capabilities to track time-variant MIMO systems. Direct application of this algorithm to the embedded input space of an MIMO system may not be feasible due to correlation over time lags that prevent the LAR algorithm from finding appropriate subsets. Therefore, we seek a novel approach to select channels using this variable selection scheme.

Assuming linear independence between input channels in MIMO systems, we may be able to apply on-line variable selection to channels instead of each tap output. In order to do that, we need to identify a variable in each channel, which can represent the temporal patterns of input time series. If we consider learning a linear MIMO system, the estimation of target signal is typically the sum of the outputs from filters at every channel. These filtered outputs can indicate the relationship between target signals and the input temporal patterns at each channel. Since on-line variable selection operates based on correlation, the filter output is hypothesized to be a sufficient variable to provide the correlation information between target signals and its input temporal pattern.

Hence, we choose the filter outputs as inputs to the on-line variable selection procedure. Then, the remaining question is how to model each filter. In modeling filters, the most important aspect to be considered is the capability of tracking nonstationary characteristics of the MIMO system since we are more interested in the selection of relevant channels than the estimation of channel parameters. It means that the learning parameters for the filters are not so important as far as the filters can sufficiently track the
nonstationary MIMO transformation. Therefore, we choose a finite impulse response (FIR) filter which has the simplest structure, and yet the maximum resolution among generalized feedforward filters [Pri93]. This maximum resolution property is important because real time channel selection will be much influenced by short-time input temporal patterns. The adaptation method for FIR filters can be many, but we choose to utilize LMS due to its simplicity and reasonable tracking performance in nonstationary environments [Hay96a]. This linear MIMO system consisting of FIR filters at each channel adapted by LMS can be further supported by the fact that the Wiener filter featuring the same topology with a simple analytical solution can estimate hand trajectory with a reasonable performance level in preliminary BMIs studies (see chapter 3 for reference).1 Hence, FIR filters adapted by LMS in real time, will yield filter outputs for on-line channel selection.

Figure 9-1 depicts the overall architecture of the real time neuronal subset selection approach. The input $x_i(n)$ at the $i$th neuronal channel for $i = 1, \ldots, M$ is filtered by an FIR filter with order of $L$, yielding the filter output vector $y(n) = [y_1(n), \ldots, y_M(n)]^T$. The autocovariance matrix $R(n)$ of $y(n)$, and the cross-correlation vector $p(n)$ between $y(n)$ and the desired hand position signal $d(n)$, are recursively estimated by (9-2) and (9-3), respectively. Then, the on-line variable selection algorithm receives $R(n)$ and $p(n)$ to yield a LAR coefficients vector $c(n) = [c_1(n), \ldots, c_M(n)]^T$. Note that some of elements in $c(n)$ can be equal to zero due to the constraint implied in LAR. The filter coefficients $w_{ji}(n)$ are updated by the normalized version of LMS using the error which is resulted as

$$
e(n) = d(n) - \hat{d}(n) = d(n) - c(n)^T y(n) \quad (9-7)$$

1 Basically, LMS converges to the solution provided by the Wiener filter in stationary environments, and additive white noise assumption.
The update of $w_{ji}(n)$ is then given by

$$w_{ji}(n + 1) = w_{ji}(n) + \frac{\eta}{\gamma + \|x(n)\|^2} e(n)c_j(n)x_j(n - i)$$

(9-8)

for the $j^{th}$ channel with the $i^{th}$ time lag, where $\gamma$ is a small positive constant and $\eta$ is a step size.

Figure 9-1. The diagram of the architecture of real time neuronal subset selection method.

It is obvious in this architecture that the constraint on the LAR procedure plays a key role in channel selection. Hence, results will be enormously affected by the selection criterion. The following will present our approaches to address this issue.

**Determination of Selection Criterion**

For the on-line channel selection algorithm, which is based on least angle regression (LAR), we need to impose a constraint for LAR to stop adding variables at some stage. Determination of this constraint impacts subset selection such that with a relatively loose constraint, the LAR procedure will yield a large subset, increasing a
chance to select irrelevant channels, and with a more strict constraint it will yield a smaller subset, possibly missing relevant channels. Hence, a careful approach to constraint determination is necessary in the selection algorithm.

We propose an approach to determine selection criterion based on the correlation between neuronal channels and desired movements. In our approach, the constraint in the LAR procedure is based on surrogate data in which the cause-effect relationship between neuronal inputs and desired response are destroyed. The constraint can be adjusted such that LAR rarely selects subsets in the surrogate data while yields reasonable selection in the original data. The surrogate data was generated by two different procedures; 1) desynchronizing input and output by delay of desired response (hand position), or 2) randomizing the phase of the hand trajectory signal while preserving its power spectral density (PSD).

In the experiments for food reaching data, however, the desynchronization data does not provide useful information about decision of constraint due to the problem of delay over continuously firing neuronal ensemble. In order to deal with this problem, further development of conditional selection is proposed. This selection criterion assesses the goodness-of-fit of FIR filters locally in time proceeding to selection in order to avoid conducting selection on irrelevant filter outputs. We believe that this conditional selection will result in more reliable neuronal subsets in real time. Details of approaches and the demonstration of experimental results are presented in the remainder of this chapter.

**Determination of Threshold in LAR Using Surrogate Data**

In the LAR procedure, variables are selected one by one at each stage along with the adjustment of the coefficients for all the selected variables. The LAR procedure can be designed to stop adding variables in the subset when a given constraint is met.
Usually, a threshold is imposed on the $L_1$-norm of the coefficient vector in LAR. However, it is very difficult to decide the proper $L_1$-norm since we do not know \textit{a priori} the optimal amount of the sum of weight magnitudes at each time instance. Therefore, we need to seek other possible ways of imposing constraints in which we can extract a practical meaning of the threshold in LAR.

One possibility is to impose a constraint based on correlation due to the fact that LAR exploits the correlation between input and regression residual. Recall that the maximum absolute correlation between inputs and the current residual decreases over stages in the LAR procedure (see chapter 4). Also, the first maximum correlation is nothing but the cross-correlation between the first selected variable and desired response since the procedure starts with all zero coefficients. At any stage in the LAR procedure, the absolute values of the correlation between all the selected variables and the current residual are intrinsically same to each other, which is the maximum over all variables. The curve of the maximum correlation over stages has the following property; if two variables which are successively selected have the similar correlation with desired response, the difference between the successive values of the maximum correlation will be small, but if the difference between the correlations of two variables with desired response is large, the maximum correlation will decrease drastically.

This property is illustrated in Fig. 9-2 where the LAR procedure in 2D input space is shown. Assuming that the input vectors $x_1$, $x_2$, and desired vector $d$ are standardized (namely, zero mean and unit variance), and $x_1$ has more correlation with $d$, the algorithm starts to move in the direction of $x_1$. It finds the coefficient $\beta_1$ for $x_1$ such that $x_2$ has the same correlation with the current residual $r_1$ as $x_1$, where $r_1 = d - y_1 = d - \beta_1 x_1$. 
Therefore, the maximum correlation reduces from $C_{\text{max}}(0) = |\mathbf{x}_1^T \mathbf{d}|$ to $C_{\text{max}}(1) = |\mathbf{x}_1^T \mathbf{r}_1| = |\mathbf{x}_2^T \mathbf{r}_1|$. By virtue of standardization, we can approximate the inner product as the angle between vectors such that $C_{\text{max}}(0) \approx \cos \theta_0$, and $C_{\text{max}}(1) \approx \cos \theta_1$, where $\theta_j$ represents the angle between $\mathbf{x}$ and $\mathbf{r}$ at the $j^{th}$ stage. In Fig. 9-2, there are two examples of the LAR procedure. The left graph shows the case when $\mathbf{x}_1$ and $\mathbf{x}_2$ have similar correlations with $\mathbf{d}$. In this case, the difference between $\theta_0$ and $\theta_1$ is small such that $C_{\text{max}}$ does not decrease much from stage 0 to 1. In the case shown in the right graph when $\mathbf{x}_2$ is less correlated with $\mathbf{d}$ than $\mathbf{x}_1$, a large difference between $\theta_0$ and $\theta_1$ makes $C_{\text{max}}$ decreases a lot. Hence, we can observe the correlation between the selected input variables and desired response through the curve of $C_{\text{max}}$.

Figure 9-2. An illustration of the successive maximum correlation over stages in the case of two variables (channels); $\mathbf{x}_j$: input variable, $\mathbf{d}$: desired response projected in input space, $\mathbf{y}_1$: regression by $\mathbf{x}_1$, $\theta_j$: the angle between the selected variables and the residual at the $j^{th}$ stage, and $\mathbf{r}_1 = \mathbf{d} - \mathbf{y}_1$.

Figure 9-3 demonstrates two examples of the maximum correlation curves. In Fig. 9-3a, there is a huge drop of the correlation between the first channel and the second channel. In this case, only the first channel will be selected with some threshold (e.g. 80).
In Fig. 9-3b, the curve does not decrease drastically until the third channel is selected.

Thus, the first three channels will be selected with the same threshold.

Figure 9-3. Examples of the maximum absolute correlation curve in LAR.

Hence, we can impose a threshold on the curve of the maximum correlation such that the procedure stops when the maximum correlation becomes less than the threshold. Then, the input channels with nonzero coefficients are selected into a subset. If the correlation with the first channel and desired response is less than the threshold, none of channels will be selected.

However, it is still an open problem to determine a threshold which might be dependent upon data. In our approach, we utilize the surrogate data to determine the threshold. We generate two types of surrogate data. The first data are composed of neuronal inputs and a delayed version of the hand trajectory. The amount of delay is chosen to be 5 second since successive reaching movements has an interval of approximately 10 second. In this case, the synchronization between neuronal firing and hand movement will be substantially destroyed. The second data consist of neuronal inputs and the perturbed hand trajectory signal. The perturbed hand trajectory signal is generated by randomizing the phase of the original signal while the PSD of the signal is
preserved to make energy unchanged after perturbation [Pal98]. A threshold is tuned in the surrogate data such that the probability of selection at least one channel at each time instance becomes very low. Then, this threshold is used for the original synchronized data of neuronal inputs and hand trajectory.

We implement this approach in 3D food reaching data (see chapter 2 for details of data). The real time neuronal subset selection algorithm is run over 3,000 samples (300 seconds). In the linear filters, there are 104 neuronal channels each delayed by 10-tap delay line. The selection algorithm starts after 100 second in order to allow LMS to adapt filter weights in the beginning. The learning rate of LMS is set such that the sum of the linear filter outputs (i.e., without subset selection) can track the hand trajectory reasonably in the synchronized data. Note that it must not be set too large such that filter outputs do not change too fast over time (if they change too fast, their correlation with desired response may not indicate the relevance of channels). In the experiment, the learning rate is set to 0.1, and the feedback parameter in the recursive estimates in (9-2) and (9-3) is set to 0.8. These parameter settings are kept for all the synchronized and surrogate data.

The experimental results with the original and surrogate data are shown in Fig. 9-4. The threshold is empirically determined based on the surrogate data (numerically, it is 80). Fig. 9-4a shows the examples of neuronal subset selection for the surrogate data while Fig. 9-4b shows the examples for the original data. The neuronal subsets are presented along with the corresponding hand trajectory (z-coordinate) for seven different reaching movements. Notice that there are significantly many channels selected in the desynchronized data. This is due to the fact that the linear filters may utilize inputs of some
Figure 9-4. Neuronal subset selection examples; (a) the de-synchronized (by 5-second delay) data, (b) the original (synchronized) data, and (c) the surrogate data with the perturbation of hand trajectory.
neurons (e.g. a neuron indexed as 94), which are known to modulate after reaching is finished, for prediction of reaching movement since these inputs are made to be aligned with reaching in time by 5-second delay (a reaching movement approximately spans 4 seconds). If we attempt to select fewer channels for the de-synchronized data by decreasing the threshold, it also reduces the selection for the synchronized data. The results with the second surrogate data with the perturbed hand trajectory are shown in Fig. 9-4c. The same threshold as above is used for the LAR procedure, with which very few channels are selected over entire data.

If we define a selection rate as the average number of selected channels at each time instance, the selection rate for the second surrogate data results in $9.1 \times 10^{-4} \pm 0.0044$. On the other hand, the selection rate for the synchronized data with the same threshold is $0.012 \pm 0.007$ (for the de-synchronized data, it is $0.011 \pm 0.007$). Therefore, for determination of threshold on the maximum correlation curve, it might be more practical to utilize the second surrogate method created with the perturbed hand trajectory since we can determine a threshold yielding a very small selection rate.

**Conditional Selection Criterion**

We have observed that there exist channels which are consistently selected during reaching movement in the de-synchronized data. In order to avoid selecting those channels in the de-synchronized data while preserving a selection rate in the synchronized data, we design an alternative approach to utilize the filter outputs prior to the selection procedure. It is based on the observation that the LMS-adapted filters do not track hand trajectory fairly if we de-synchronize the data. It implies that the filter outputs used for subset selection may not contain adequate information about individual neuronal contributions to model output. Hence, at every time instance we can proceed in two steps;
in the first step we assess how much the filter outputs are correlated with desired
response then proceed to the next step of channel selection only if the correlation measure
exceeds certain threshold. Note that this threshold is different from the one that we have
used for the maximum correlation in the LAR procedure. Adjustment of this threshold
may reduce the selection rate in the de-synchronized data with little deterioration of
selection in the synchronized data. Figure 9-5 demonstrates examples of the sum of filter
outputs in the synchronized and the de-synchronized data.

Figure 9-5. Demonstration of filter outputs before subset selection; (top) synchronized
data, (bottom) de-synchronized data.

The correlation between filter outputs and desired response is estimated as follows.

The correlation between the sum of filter outputs and desired response is measured at
every time instance. We estimate the correlation by RLS type recursion as used for on-line selection algorithm. The correlation estimate at time instance \( n \) between the sum of filter outputs \( y(n) \) and desired response \( d(n) \) is given by

\[
C_{yd}(n) = \mu C_{yd}(n-1) + \frac{y(n)d(n)}{\sqrt{y^2 p(n)d^2 p(n)}}
\]  

(9-9)

where \( \mu \) is another forgetting factor for this recursion of correlation (set to 0.95 in the experiment). Note that the product of \( y(n) \) and \( d(n) \) is normalized by the square root of local signal power estimates in order to avoid biasing of correlation measure to a large magnitude of \( d(n) \). The local (in time) signal power estimates, which are denoted by \( y^2 p(n) \) for filter output and \( d^2 p(n) \) for desired response, are computed by the same recursion,

\[
y^2 p(n) = \mu y^2 p(n-1) + y^2(n)
\]

\[
d^2 p(n) = \mu d^2 p(n-1) + d^2(n)
\]

(9-10)

If \( C_{yd}(n) \geq \theta \) for certain threshold \( \theta \), the on-line selection procedure is run on the filter outputs. If \( C_{yd}(n) < \theta \), we make all the LAR coefficients \((c_j(n) \text{ in Fig. 9-1})\) equal to zero, yielding an empty subset. The threshold \( \theta \) is determined empirically such that few channels are selected in the de-synchronized data over entire data \((\theta = 0.7 \text{ in the experiment})\). The other threshold for the maximum correlation in the LAR procedure is kept same as above. Note that in this case the filter coefficients may be kept adapted by the standard LMS algorithm even when its output is not selected. The reason is that some filter coefficients can be kept unchanged for long time, losing tracking ability due to conditioning on selection. Hence, it would be preferred to adapt every filter coefficients at every time in order to track nonstationarity. This means that every coefficient is updated by (9-8) with letting \( c_j(n) = 1 \) for all \( j \).
The neuronal subset selection results for the same seven movements as above are shown in Fig. 9-6 for the de-synchronized data and the synchronized data. The selection rates are $0.006 \pm 0.006$ for the synchronized data, and $0.001 \pm 0.003$ for the de-synchronized data. Compared with Fig. 9-4, the subset selection in the de-synchronized data becomes very sparse, whereas similar neuronal subsets are selected in the synchronized data. For the surrogate data with the perturbed hand trajectory, no subset is selected in this case due to the additional constraint on the filter outputs. These results demonstrate that we can determine thresholds with the surrogate data combined with the condition on the correlation between filter outputs and desired response such that neuronal subsets are selected only in the synchronized data.

**Experiments of Neuronal Subset Selection**

With the conditional selection criterion as described above, now we implement the neuronal subset selection method in BMIs. In order to ensure the robustness of the selection results to initial conditions, we run the simulation 50 times to obtain multiple realizations of selection. Then, we define a selection vector as

$$s(n) = [s_1(n), s_2(n), \cdots, s_M(n)]^T$$

$s_j(n) = 1$ if the $j^{th}$ channel is selected, and $s_j(n) = 0$ otherwise. The average vector of $s(n)$ over 50 realizations are computed for every $n$. Figure 9-7 depicts these average vectors for the same seven movements as above. The results show that if the $j^{th}$ channel is selected, the average of $s_j(n)$ becomes very close to 1. It reveals that a subset of neurons is consistently selected with different initial conditions for the same movement.
Figure 9-6. Neuronal subset selection conditioned by the correlation between filter outputs and desired response; (a) de-synchronized data, and (b) original synchronized data.
Figure 9-7. Demonstration of the robustness of the algorithm to initial conditions.

In Fig. 9-8, the tracking performance of our approach is compared with that of the straight linear MIMO systems trained with LMS for the best tracking. In other words, it demonstrates the effect of the on-line channel selection algorithm on tracking performance. The sample outputs of both our MIMO system with on-line channel selection and the straight linear MIMO system are displayed on top of the actual hand trajectory in z-coordinate. Although the statistical measurements of performance in terms of the mean squared deviation and the misadjustments [Hay96a] must be conducted, which will be pursued in future studies, we can clearly see from this figure that our tracking system identifies the peaks of hand trajectory a lot better than the straight linear system. Note that the parameter settings in the LMS algorithm for both systems are identical for the purpose of fair comparison. This superior tracking performance of our system may result from the fact that additional spatial filtering by the on-line channel selection algorithm optimally combines filter outputs to reduce the instantaneous error.
between desired response and the final output. Also, the sparse set of coefficients estimated by the on-line channel selection algorithm may play a role of adjusting the update of weights for individual linear filters which are once adapted by LMS with a constant rate.

Figure 9-8. An example of the outputs of two tracking systems with (solid line), and without on-line channel selection (dashed line).

It is often necessary to find a subset of neurons that are relevant to current movement (of any dimension) without separation into each coordinate. For this purpose, we need to perform selection for individual coordinates and combine the selection results into one since the on-line variable selection algorithm is currently developed only for single dimensional output. Therefore, after obtaining selection vectors \( s(n) \) from every coordinate, we simply perform Boolean OR operation with those vectors to yield a combined selection vector, by assuming \( s(n) \) as a Boolean vector (by virtue of the fact that \( s(n) \) consists of 1 or 0). It means that if a channel is selected for at least one coordinate of hand position, it joins the selected subset. Figure 9-9 demonstrates the
combination result of neuronal subsets for all three coordinates of hand position. It is obvious that more channels are selected with combination compared to Fig. 9-6b. With this combination, we can obtain single representative of neuronal selection for a given movement instead of examining individual selections for each coordinate of hand trajectory.

Next, we investigate if the relationship between neuronal population and behavior varies over time by the analysis of neuronal subsets. In order to account for this variation in time, we execute neuronal subset selection over a long period of data (2,000 seconds). Then, we analyze neuronal subsets in the early part of the data, and the ones in the late part of the data. The resulted subsets and corresponding hand trajectory at z-coordinate are depicted in Fig. 9-10. There are interesting observations in these subsets; there are

![Figure 9-9. Neuronal subset selection for all three coordinates of food reaching movement.](image)
neurons which consistently contribute to movement over time such as 5, 7 and 93. But, some neurons, which are selected in the early part of the session, do not seem to be involved in later movements (e.g. 70). Also other neurons including 23 and 71 are not selected in the early part of the session, yet join the selection for the late part of movements. It is also interesting to see the transition of contribution from neuron 70 to neuron 71 over time since the activities of those neurons are collected from adjacent

Figure 9-10. Neuronal subset selection over 2,000-second data; (a) subsets in the early part, and (b) subsets in the late part of the data.
electrodes in PMd. These observations demonstrate that the real-time neuronal subset selection method can provide a very useful tool to understand the nonstationary properties of neuronal population associated with behavior. Now, we apply the neuronal subset selection method for another BMI dataset; 2D target reaching of Aurora. The thresholds used for conditional selection criterion are empirically determined such that the selection rate with the de-synchronized data is much smaller than that with the synchronized data. The de-synchronized data is generated by time delaying the hand trajectories by 10 seconds. With certain threshold (numerically, $\theta = 0.3$), the selection rates turn out to be $0.002 \pm 0.006$ for the de-synchronized data, and $0.015 \pm 0.009$ for the synchronized data, respectively. The other threshold on the maximum correlation curve is determined empirically such that the selection rate with the other surrogate data generated by randomizing the phase of hand trajectory is much smaller than that with the original BMI data.

Real time neuronal subset selection with these settings of threshold is performed on 1,600-second long data (16,000 samples) consisting of 185-channel neuronal bin counts and 2D desired hand positions. The FIR filter at each channel has an order of 10. The LMS algorithm is applied to adapt FIR filter coefficients in real time. A Boolean selection vector $s(n)$ for x-, and y-coordinates are combined at every time instance by OR operation. The results of the combined subset selection are shown in the bottom panel in Fig. 9-11. The subsets correspond to five sample segments of the entire hand trajectory, which exhibits a similar trajectory as illustrated in Fig. 9-12. Inspecting neuronal subsets over different segments, we can obtain neurons that are consistently selected such as 69,
80, 84, 92, 99, 108, and 110. However, there are a number of neurons that are selected only in particular segments such as 45, 54, 67, 149, and so on. Most of selected neurons are collected from M1 area. Yet, neurons from PMd area become parts of subsets in the last two segments (late part in the dataset).

Figure 9-11. Neuronal subset selection for a 2D target reaching BMI.

Figure 9-12. 2D hand trajectories in five sample data segments selected in Fig. 9-11.

2 These neurons are selected in at least three segments.
Discussions

With the proposed conditional selection criterion, the on-line channel selection algorithm is activated only when the correlation between the LMS-adapted filter outputs and desired response is larger than certain threshold. However, there must be moments at which some neuronal channels are significantly relevant to hand movements even when the sum of outputs is not correlated with desired response due to noise. Also, the imposing constraint on the maximum correlation curve in the LAR procedure may not be an optimal choice. Hence, there are still a lot of options for selection criterion to enhance neuronal subset analysis in real time.

It is noteworthy that neurons that are selected by subset selection match with those with NMF and the sensitivity analysis. For instance, neurons indexed by 5, 7, 23, 71, and 93 are observed in NMF basis vectors, or in the top-ranked group of neurons assorted by the sensitivity analysis, for food reaching. Also, neurons indexed by 54, 69, 80, 84, 92, 99, 108, 110, 149, and 167 are observed in NMF basis vectors, or in the top-ranked sensitivity group, for target reaching. These comparisons show us that the subsets selected in real time are not different from neuronal subsets determined by stationary methods. However, the advantage of real time subset selection is the capability of detecting time-varying changes of composition of subsets in a nonstationary environment, and under these conditions, the fitting will be greatly improved.

One important question is to extend this data analysis tool that requires the desired signal (behavior) for real time subset selection in BMI during the testing phase when no desired response is available. This is a non-trivial problem since desired response is usually not available after the model is trained. Hence, the subset cannot be found with the correlation between neuronal channels and desired response in the evaluation mode.
However, if we can somehow modify the selection process such that selection can be executed without target information in real time, it will be enormously useful for decoding models in BMIs. One plausible idea is to find the relationship between each filter output and selection; for which characteristics of output, the channel is selected. It may require detection of output patterns or classification in output space. In any case, this will be an exciting topic to pursue in future study.

Although we have demonstrated subset selection in this chapter, more rigorous analyses must be conducted to quantify the results of subset selection. These analyses may involve from the fundamental statistical analysis of subset to the advanced probabilistic approaches to investigate the synchronous activity of neural ensemble. Among such analytical approaches, a data mining technique developed for determining the synchronous co-activation subset of multi-channel input [Bou04] is of special interest since it provides a methodology to quantify the reproducibility of co-active patterns in multi-channel data. It also forms a series of Boolean vectors from the particular activity events in each channel in order to automatically extract the synchronous co-activity patterns. Since the neuronal subset selection data can be dealt as a series of Boolean vectors as depicted in previous section, this methodology may provide an appropriate way to extract very useful information about the synchronous activity of neural ensemble combined with subset selection. This topic must be covered in the following future investigations.

Albeit there is a wide range of ways to perform the analysis in the neuronal subsets, we can glimpse the characteristics of the selected neuronal subsets by relatively straightforward quantitative evaluations. Here we demonstrate a few examples for such
quantification using food reaching data. Since we are only interested in subsets selected during movements, we divide the entire subset data into individual segments for each reaching movement. This procedure results in 149 segments corresponding to each reaching movement in a training dataset.

The first example describes the selection of individual neurons for each movement as shown in Fig. 9-13. If a neuron is selected during a given movement at least for two consecutive time instances, the neuron is determined to be in the set of selection for that movement. In this graph, we can observe that which neurons are consistently selected over many reaching movements. For instance, neurons indexed as 5, 7, 23, 29 and 93 are shown to be selected for overall movements. These neurons also exhibited the large relations with the hand movement in the sensitivity analysis or the NMF bases (see table 8-1). On the other hand, neurons indexed as 19, 45, 71, and 84 are partially selected for some movements. Especially, the neuron 71 is mostly selected in the movements.

Figure 9-13. Selection of individual neurons over a series of reaching movements.
occurred in the late part of the data. This neuron has been discussed in previous sections in order to reveal the nonstationarity of neural activity. It is now clarified by this quantification of subset selection. Notice that these neurons were also identified by the sensitivity analysis or NMF. However, with those methods we could not discern the temporal characteristic of the relation for individual neurons, which is now feasible through this real time analysis.

In the second example, we evaluate the distribution of the size of subset in each movement as depicted in Fig. 9-14. For each movement, we count the number of bins for which a subset contains \(k\) neurons for \(k = 1, \ldots, 8\). The bins for which the number of neurons exceeds 8 belongs to the group of bins with 8 neurons selected. Then, we display the counting results in color map per movement. In this figure, we can see the tendency of increasing coactivity as time increases since the number of bins with more than one neurons being selected tends to increase in the late movements. This observation may give us a clue for very important aspects of the behavior of neural ensemble in motor cortex with respect to reaching movement (for instance, the increasing co-activity of neurons during training for a particular task). However, the investigation for this subject must be conducted in more thorough way such as the statistical procedure in [Bou04]. It must also be accompanied by neurophysiologic investigations. Nevertheless, this will be a very attractive research topic in BMIs.

Finally, in order to ensure the validity of neuronal subset selection, the misadjustment of our MIMO system using on-line channel selection is compared with that of the straight linear system. Figure 9-15 shows the average misadjustment computed for each movement. In this figure, the MIMO system with on-line selection exhibits
superior performance to the normal MIMO system for most movements. This result is consistent with the demonstration of tracking performance in Fig. 9-8.

Figure 9-15. Comparison of the average misadjustment per movement between the standard MIMO system learned by LMS and the MIMO system with on-line channel selection.
Conclusions

Inspired by an excellent performance of the Wiener filter algorithms in the estimation of movement parameters from the activity of large (100–200 cells) neuronal ensembles, we conducted an extensive comparable study of MIMO filters in BMI design. As test data, we used two datasets, each collected in different experimental BMIs, one in a monkey reaching for food in 3D space, and the other one in a monkey reaching a visual target in 2D space. Although in certain comparisons, different models had very similar performance quality, we anticipate that with the development of BMI field and especially with the increase in the number of simultaneously recorded neurons, some of these modeling ideas will find important applications. For the present datasets, all the MIMO filters including the standard Wiener filter performed very well in spite of the large number of degrees of freedom (over 3,000 parameters) and the absence of regularization. The major reason for such high performance quality may be due to an excellent quality of the neuronal recordings. Multiple microelectrode arrays were strategically implanted in cortical areas known to be associated with arm and hand movements. In addition, special care was taken to keep experimental conditions controlled and restricted to specific task requirements. It still remains to be studied how the linear models perform as the range of motor performances and experimental conditions becomes more complex.

Notwithstanding a good performance of non-optimized Wiener filters for these datasets of 100-200 spatially tuned neurons, we showed that the amount of input data
could be reduced. The number of parameters of the linear model was decreased using two different approaches for pruning in time and in the space of the electrodes. In the time dimension, we used gamma delay operators instead of ideal delays to decrease the number of coefficients while spanning the same memory depth (although with a coarser resolution). The gamma model produces statistically better models when compared to the Wiener filter.

Pruning in electrode space is achieved using two different strategies: selecting important channels and using regularization methods to control complexity. The selection of channels with PCA (input neuron information) does not perform well, however, a combination of PCA and PLS that chooses subsets of neurons based on their importance in the joint (input and desired signals) space is able to statistically outperform the conventional Wiener filter in both tasks. Likewise the weight decay regularization also statistically outperforms the Wiener filter. However, the regularization parameter must be appropriately selected in cross-validation; otherwise the performance is very brittle. Therefore, we conclude that the tools of regularization theory are an asset for optimal modeling in BMIs, but the improvements are smaller than expected in spite of being statistically significant.

Comparison of the performance of nonlinear versus linear models showed better performance of nonlinear model for one dataset (food reaching), but not for the other (target reaching). Nonlinear models significantly outperformed the linear counterparts for the food reaching task, mostly due to their ability to follow better the non-movement (hand at rest) portions of the desired response. This is due to their ability to “shut-off” parts of the network by virtue of nonlinearity. However, in the target reaching task where
the hand is almost always moving, the performance was very similar, being statistically indistinguishable from the Wiener filter. Given the complexity of brain networks and no a priori reason for them to have linear properties, this was unexpected, and may reflect the fact that it is harder to train nonlinear models to the same specification of the linear ones. Or simply that due to the large input space of BMIs finding a linear projection space of reduced dimension (2D or 3D) is sufficient when performance is the only metric. In addition, one would expect a better performance in a nonlinear model when it matches in some ways the performance of the real brain network, otherwise it would falter. Linear models on the other hand already incorporate well-known properties of cortical neurons, such as directional tuning (typically described by a cosine function), sensitive to position, velocity and force.

The challenge aroused from performance saturation of both linear and nonlinear models led us to view BMI signal processing in a different angle. With the redundant representation of neuronal firing activity through the multiresolution analysis of spike trains, the performance level of a simple linear model (with regularization) was increased. Although the extent of performance improvement was marginal with a simple reconstruction of neuronal input space, these experiments showed us the importance of the encodings of neural information for BMI models. This preliminary result will lead us to seek a congruent set of encoding basis for neural information from which a decoding model can easily find mapping to behavior.

We postulate that the nonlinear topologies may have practical advantages when BMIs are implemented in real time digital signal processors. Work reported in [San03b] shows that when memory constraints and clock cycles are taken into consideration,
RMLP requires a smaller computation bandwidth and resources than the FIR filter trained with NLMS. However, training of the RMLP is still more complex than the NLMS algorithm, so further work to find nonlinear topologies that train faster should be sought. The successful implementation of echo state networks as a decoding model for BMIs gives one possible direction for addressing this issue. In terms of the regularization techniques, the gamma model and the weight decay can easily be implemented in DSPs, but the subspace Wiener filters require a substantial increase in computation. Therefore, further work to simplify these algorithms should also be pursued. In terms of deployment, a BMI with 100 channels to predict 2 or 3D hand trajectories based on the regularized NLMS filter can be implemented in real time in a small Texas Instruments C33 WiFi board recently developed by our group.

A comment regarding prediction performance of these algorithms in terms of correlation coefficient (CC) is in order. The CC of all these algorithms is capped at 0.8 for the food reaching tasks and 0.7 for the target hitting tasks. It is important to investigate if this limit is related to missing data (only the tiny percentage of the motor cortex neurons is probed) or if it is the intrinsic spatio-temporal nonstationarity of the data that is not properly captured by this class of models that learn based on stationarity assumptions. Another important issue that should not be forgotten in the design of better BMIs is how to effectively include neurophysiology knowledge both in the filter topologies and in the cost functions.

Besides well-earned clinical applications, what experimental BMIs newly bring to researchers is the opportunity to investigate the functional organization of neural ensemble associated with behavior in real time. And, this investigation is often coupled
with the fitted models such that we can gain a wealth of information from model parameters. With this respect, we stepped into the development of engineering solutions for the analysis of neural systems and their relation with motor functions. In our works, two methodologies were proposed to provide valuable analytical tools, including pattern determination using non-negative matrix factorization (NMF), and a real time neuronal subset selection algorithm. Processed by NMF, spatio-temporal patterns of neuronal ensemble could be effectively represented in NMF basis vectors, and the contribution of each pattern to motor parameters could be estimated. One of intriguing aspects of this analysis is that only by simple factorization of a neuronal firing count matrix we could obtain the information of spatio-temporal characteristics of neuronal populations. This arouses especial interest since no one has demonstrated a way of ascertaining synchronization of a group of cells in such a simple fashion, without analyzing each cell property. Although we are not at a stage of full understanding how NMF can find synchronization in very complex neuronal firing data, further apprehension of NMF and BMIs will lead us to consolidate this tool for many neuroengineering applications.

However, as other current solutions, NMF is limited in the case of stationary environment since it factorizes a block of data. Therefore, the second attempt was made to utilize the adaptive filter coefficients for probing time-varying changes of neuronal contributions to movements. In order to overcome the difficulty in tracking a huge MIMO system with the standard on-line adaptive algorithm such as LMS or RLS that is governed by constant parameters, we proposed to utilize an on-line variable selection scheme to linear filters. With a proper setting of selection criterion, the on-line selection algorithm could spot a subset of neurons that was correlated with a present part of
movement. The profile of the selected subsets also matched with preliminary methods including the sensitivity analysis and the cellular tuning analysis. Moreover, due to its real time operation, this algorithm could detect the change of neuronal contributions over time. With further calibration of the procedure, we believe that this analytical tool will be a useful probe for the investigation of neural system analysis based on the BMI setups.

**Future Works**

We will pursue research in BMIs in two main thrusts; the design of decoding models, and the analysis of neuronal ensemble coding with behavior. As browsing a wide range of linear and nonlinear models from adaptive and statistical learning theories to obtain better fitted models for BMIs, we have experienced that the existing real time modeling frameworks might reach some limitations. Although we must admit that this limitation may come from the extremely sparse sampling of neuronal activity among millions of motor cortical cells, there seem to be still a good deal of potentials to design a better model. This will also become plainer as more complex and diverse do the experimental paradigms and the goals of BMIs. Hence, we will continue to seek chances to build more suitable BMI models. What now we see as feasible approaches is based on our preliminary studies; further development of mixture of experts, and an adaptive system accounting for nonstationarity. A nonlinear mixture of linear models (NMCLM) has demonstrated that with a proper mixing function, we could improve performance. The defective performance of this model for a continuous 2D target reaching BMI might be probably due to lack of an adequate mixing criterion. A different approach based on the switching Kalman filter model has also shown that a mixture of local models could boost performance [Wu04]. Hence, we will step forward in this direction aiming at finding appropriate localization and establishing a proper mixing function, thus boosting
generalization and accuracy of the mixture. Although we apprehend that brain activity and movement generate completely nonstationary signals, we have not been able to create models which can track time-variant systems. Only close approaches has been based on state estimation using Kalman filters [Wu03, Wu04], or a recursive estimation of tuning properties for the population vector coding [Tay02]. A major difficulty in this type of modeling is that we have to track changes occurring in the joint space of input/output, which is not feasible after training without information of desired signals. Hence, the possible alternatives may be to utilize a database built during training stage in suitable ways. With a proper extraction methodology for nonstationary characteristics in the joint space, and the precise construction of pavement for model parameters, we might be able to continuously update a parameter vector after training is finished. This research topic is now only beginning in the adaptive learning theory field, and further developments will lead us to yield a model for nonstationary environments in BMIs.

We have started to engineer signal processing tools for probing neural systems in experimental BMI setups, including pattern determination using NMF, and real time subset selection. Although the applications of these tools to brain research seem to be promising, there are still remaining issues to be solved. As for NMF, we need to fully understand how the NMF learning algorithm captures repeating patterns in the input matrix. Also, it has been shown that for a 2D target reaching BMI we had to employ the multiresolution representation (with relatively larger scales) of firing rate in order to extract repeating patterns by NMF. This leads us to enhance NMF to be effective for the data with a complex structure. In the real time neuronal subset selection method, we first have to establish a way to verify that the selected subset makes a biological sense. We
have investigated the overall distribution of neuronal subsets by comparing it with the
results of other methods, and demonstrated the compatibility between the subset
distribution and the assorted neurons from the sensitivity analysis and NMF. However,
temporally local selection resulted in rather various subsets for the repeated reaching
movements. These observations pose a question; is time-varying subset selection caused
by the nonstationary aspects of neuronal ensemble, or by the stochastic properties of the
adaptive algorithm? To address this question, we need to form a solid methodology to
minimize the chance that the subsets are generated by the stochastic nature of the
adaptive mechanism. Another issue in our approach is that selection is based on the
assumption of linear relationship. However, there is little chance that the true relationship
between neuronal firing activity and motor parameters is linear. Hence, if we can design a
model to track time-variant nonlinear system, it will find more convincing subsets that
are not restricted by the linearity.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Sung-Phil Kim was born in Seoul, South Korea. He received a B.S. in the Department of Nuclear Engineering from Seoul National University, Seoul, South Korea, in 1994. From 1994 to 1997, he worked for the Network Solution and Sales Supports team in Comtec Systems, Inc., Seoul, South Korea. In 1998, he entered the Department of Electrical and Computer Engineering at University of Florida in pursuit of a Master of Science. He joined the Computational NeuroEngineering Laboratory as a research assistant in 2000. He also received an M.S. in December, 2000, from the Department of Electrical and Computer Engineering at the University of Florida. From 2001, he continued to pursue a Ph.D in the Department of Electrical and Computer Engineering at the University of Florida under the supervision by Dr. Jose C. Principe. In the Computational NeuroEngineering Laboratory, he has investigated decoding models and the analytical methods for brain-machine interfaces. His research is funded by the Defense Advanced Research Projects Agency, and is part of a joint research project with Duke University, State University of New York, Massachusetts Institute of Technology, Plexon, Inc., and the University of Florida.