Combining Neural Population Recordings: Theory and Application

William Bishop

September 2015
CMU-ML-15-106

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Byron Yu, Chair
Maneesh Sahani
Rob Kass
Geoff Gordon

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Copyright © 2015 William Bishop

This research was funded by: the National Institutes of Health awards T90DA022762, R90DA023426, R01HD071686 and R01NS105318; the National Science Foundation award BCS1533672; a grant from the Simons Foundation; a grant from the Craig Neilsen Foundation; and a grant from the Pittsburgh Life Sciences Greenhouse Facilities & Opportunity Fund.

Keywords: neural recordings, stitching, electrophysiology, optical methods, dimensionality reduction, matrix completion, factor analysis, positive semidefinite matrices, brain-computer interface, learning, multi-area communication

Abstract

Modern electrophysiological and optical recording techniques allow for the simultaneous monitoring of large populations of neurons. However, current technologies are still limited in the total number of neurons they can simultaneously monitor, the brain areas they can be deployed in and in their ability to stably record from a given set of neurons. This thesis develops machine learning theory and algorithms which address these limitations by combining neural population recordings across time and space.

We combine neural recordings by leveraging two types of structure present in neural activity: clustering and low-dimensional structure. We first use clustering in neural activity to develop self-recalibrating brain-computer interface (BCI) classifiers. Modern BCI classifiers require daily recalibration, which may present a burden to future patients in a clinical setting. We show that by exploiting clustering in neural activity that arises in a classification setting we can develop self-recalibrating classifiers capable of stable performance over 26 and 31 days of prerecorded data without any supervised recalibration or retraining.

In the remainder of the thesis, we develop theory and algorithms to exploit a second type of structure which commonly appears in neural recordings: low-dimensional structure. We first present novel matrix completion theory and an algorithm applicable to learning the full covariance matrix for a population of neurons recorded in overlapping blocks. The presented theory applies in general to completing low-rank symmetric positive semi-definite (SPSD) matrices, and we present sufficient and necessary conditions for this problem. We also show that matrix completion is possible under our sufficient conditions via nuclear norm minimization, a well known matrix completion technique. These conditions are notable as they apply when entries of a matrix are observed in a deterministic, structured manner and make no appeal to incoherence.

We then extend our methods to problems beyond that of simply learning the joint covariance structure for a population of neurons. We describe how factor analysis (FA) models can be fit to the joint activity of an entire population of neurons recorded in blocks. Once fit, these models can be used to infer the low-dimensional state of the entire population of neurons at different moments in time, even when neural activity from only a subpopulation is observable at any particular point in time. Building from our matrix completion theory, we present an intuitive set of necessary conditions for fitting FA models in such settings.

After developing the basic theory for fitting FA models in such settings, we validate our techniques in three application scenarios. First, we develop a self-recalibrating BCI regression algorithm, which is capable of maintaining steady performance in the face of simulated electrode instabilities. Second, we demonstrate that the progression of the low-dimensional state of a population of neurons can be tracked in a learning setting, even when it is impossible to record from a fixed set of neurons over the entire period of interest. Finally, we apply our methods to identify subspaces which are potentially important for communication in the activity of neural populations in two brain areas when it may only be possible to record from one neuron at a time in one of the brain areas.

Acknowledgments

I would like to thank the many people who have supported me before and through my PhD. I would like start by thanking Byron Yu and Rob Kass, my advisors. Byron, who I was primarily advised by, has provided tremendous support. The environment of the lab he fostered, attention to detail, insights provided, resources he made abundantly available, time he poured into my work and commitment to my development has been tremendous. I have likewise benefited greatly from Rob's statistical insight, guidance, perspective and career mentorship. I would like to thank both Rob and Byron for their patience, support and belief in me during the difficult period of my PhD and their encouragement to follow my interest in theory.

I have benefitted greatly throughout the years from the mentorship and support of many others, and this certainly includes all those I worked with as an engineer on the Revolutionizing Prosthetics 2009 (RP2009) project during my time at The Johns Hopkins Applied Physics Lab (APL). James Beaty, my manager and certainly early career mentor, trusted a new college graduate with tremendous opportunities, allowing me to discover a passion for neuroscience and machine learning. I would like to thank the entire APL leadership team who brought me onto the RP2009 project, invested in my development as a young engineer and researcher and supported me completely as I chose to pursue a PhD. More than he may know, my research path has been greatly influenced by Krishna Shenoy who, beyond making his and his lab's time and resources available to collaborate with and guide a new engineer, has provided so much inspiration about what a career in research could look like. Marc Schieber provided perspective as I was contemplating returning to graduate school, and I benefited greatly from the time spent in his lab on the RP2009 project.

My time during my PhD has been similarly enriched with collaboration and from the support of many people. Aaron Batista and all those in his lab have provided many rich conversations and have been unbelievably generous in their willingness to collaborate. Conversations with Steve Chase have provided many useful insights. Marlene Cohen and Douglas Ruff welcomed me to spend time in their lab and gave me my first introduction to the study of the visual system. Adam Kohn made recordings collected in his lab available for a portion of the work in this thesis. Aarti Singh provided guidance as I delved into the problem of matrix completion, and Geoff Gordon has never ceased to provide a wealth of ideas and directions for the machine learning aspects of my work. Maneesh Sahani gave his time as a member of my committee and provided helpful questions and suggestions regarding my research. Finally, Diane Stidle worked tirelessly to support and look after me as well as all the other machine learning students during my PhD.

There are many friends and family to thank as well. I am deeply grateful for the friendship of Aadi Ramdas and Leila Wehbe. I would like to thank my parents for their prayers, love, support, belief in me and everything they have provided. I would like to thank my brother Danny and sisters Rosemary and Caroline for their love and support through the years. I know I am truly blessed to have such a wonderful family. Finally, I would like to thank all those in the Agapé Churches for all they have given, their prayers and their love. My life has been wonderfully changed because of what God has done through them.

Contents

Chapter 1

Introduction

Modern electrophysiological [1] and optical recording [2] techniques allow for the simultaneous monitoring of large populations of neurons. This has opened up new possibilities for the clinic and avenues of exploration in basic science. In the clinic, state-of-the-art brain-computer interfaces (BCI) have allowed a small number of patients in clinical trials to control multi-degree of freedom prosthetic limbs (e.g., [3, 4]). In basic science, analyses of neural population activity has led to new insights for motor control [5], decision making [6] and sensory perception [7].

Nonetheless, our ability to record simultaneously from large numbers of of neurons in any area of the brain stably over time is still limited. This constrains the type of scientific questions we can ask and the practical utility of BCI. In this thesis we propose to use a common property of neural population activity, structure, to develop computational tools to ask scientific questions and make clinical progress in the face of these limitations. Specifically, we will show that by leveraging structure in neural activity we can combine recordings from overlapping populations of neurons made at different times. This will allow us to address stability issues encountered with BCI and analyze scientific questions using the activity of an entire combined population of neurons.

1.1 Structure in Neural Activity

Throughout this thesis, we will measure neural activity using counts of action potentials. Neurons are cells in the nervous system which compute and encode information by sending trains of action potentials to one another. As illustrated in Fig. 1.1A, action potentials are distinct spikes in the membrane voltage of a neuron. We can measure the activity of a population of neurons in a given window of time by counting the number of action potentials produced by each neuron during a window of interest. These counts of action potentials are often referred to as spike counts. When we divide the number of action potentials by the duration of the window we count them in, we arrive at an estimate of neural firing rates.

Neural activity, when measured with spike counts, often possesses structure. In this thesis, we leverage two types of structure to combine recordings of neural activity. The first type of structure is clustering and is illustrated in Fig. 1.1B for a population of three neurons. Clustering can come about when the brain performs a set of distinct computations. For example, activity in motor cortex may take on a clustered structure in an setting where a subject repeatedly makes reaches in a small number of directions. In this case, each cluster would correspond to a reach direction.

The second type of structure is low-dimensional structure and is again illustrated for a three neuron example in Fig. 1.1C. Neurons are connected to one another in a network, so that action potentials produced by one neuron influence the production of action potentials by another. In many settings, the patterns that neural activity move in can be characterized and can be used to describe a low-dimensional space. As reviewed next, low-dimensional structure has been found in many parts of the brain and in many experimental settings.

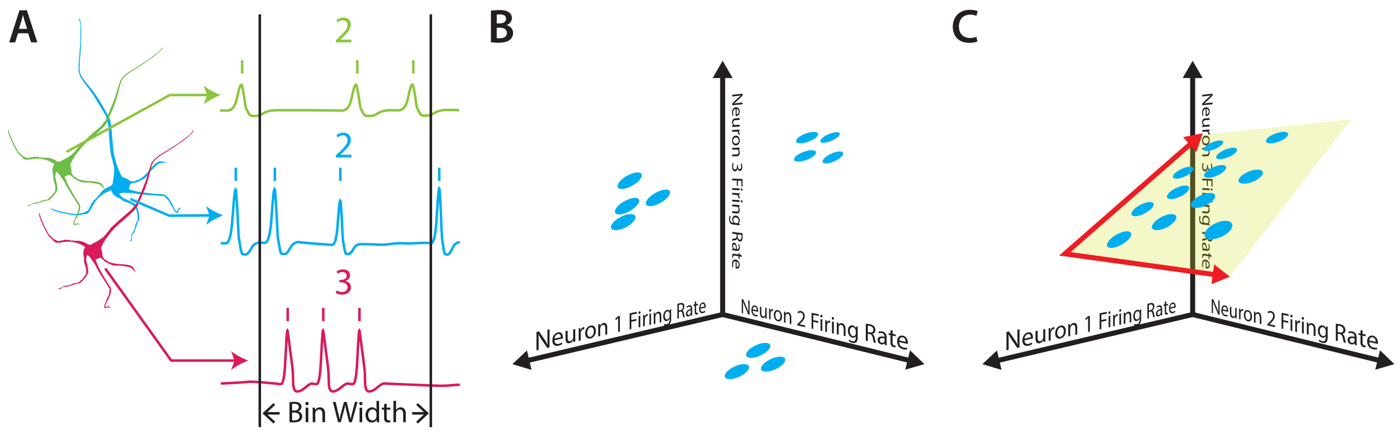


Figure 1.1: (A) In this work we measure neural activity be counting action potentials produced by neurons in windows of interest. If we divide the number of action potentials by the width of the window (or "bin") in which they are counted, we arrive at an estimate of firing rate. (B) An example of clustered structure in the activity of three hypothetical neurons. Each dot represents neural activity measured in one window. (B) An example of low-dimensional structure in the activity of three hypothetical neurons. Even though each dot represents the joint activity of three neurons, neural activity falls in a two-dimensional plane.

1.1.1 Low-Dimensional Structure in Previous Studies

There have been many studies which make use of low-dimensional structure in neural data. While it would be impossible to list all here, we highlight some notable examples from different systems. See also [8] for a recent review.

Multiple studies in motor cortex have analyzed population activity with dimensionality reduction methods. Yu et al. [9] develop a technique to extract smooth time courses of low-dimensional neural state on a single-trial basis and visualize neural activity converging during the planning period of a motor task. Churchland et al. [5] analyze trial averaged activity and observe rotational dynamics in neural activity, which may provide a more parsimonious explanation of the activity of neurons in motor cortex than what has been achieved with more traditional tuning curve based approaches. Kaufman et al. [10] analyze neural activity patterns in motor cortex during planning and movement periods, and find that neural activity evolves largely in two orthogonal subspaces between the two periods, suggesting a possible means for neural activity in a planning period to evolve without driving limb movement. Ames et al. [11] visualize low-dimensional representations of neural activity in tasks in which a monkey must make changes in planned arm movements. They find evidence that neural activity need not traverse the same region of state space when arm movements are planned well in advance or re-planned in response to changing task demands. Finally, recent work by Sadtler et al. [12] used a BCI task to demonstrate that low-dimensional structure present in motor cortical neural activity could be used as the basis for predicting the learnability of a BCI mapping.

In olfaction, Mazor and Laurent [7] construct low-dimensional trajectories of neural activity from projection neurons recorded in the locust antennal lobe to examine network dynamics during the processing of odor. Hallem and Carlson [13] applied principal component analysis to the responses of 24 types of odorant receptors presented with 100 odors and visualized clustering of the responses by chemical class of the presented odorants.

In vision, Cohen and Maunsell [14] record simultaneously from populations of neurons in visual area V4 in an orientation detection task. By projecting neural activity onto a one dimensional axis oriented to discriminate attention conditions, they find meaningful differences in the neural population response that were predictive of the successful detection of an orientation change.

Finally, in prefrontal cortex, Brendel et al. [15] develop a dimensionality reduction technique designed to find directions in neural space which cleanly represent stimulus or choice information and find directions in PFC neural activity that represent, time, frequency and choice in a two-frequency stimulus discrimination task. Mante et al. [6] similarly use dimensionality reduction to visualize low-dimensional neural trajectories during a context dependent decision task in which a monkey must chose the direction of an upcoming saccade based on the majority direction or color of a random dot display. They find no evidence that irrelevant stimulus information is filtered out in early stages of processing, but instead find that dot direction, color and the choice are represented simultaneously but along different directions in the space of PFC neural activity. The ability to discriminate stimulus information and choice is largely only possible at the population level, as individual PFC neurons are well known to often respond to a mixture of both stimulus information and choice.

1.2 Limitations of Current Recording Technologies

Having reviewed how we measure neural activity and the type of structure we will leverage in this activity, we now briefly review limitations in current recording techniques. In this thesis, we will seek to develop a set of computational tools to address these limitations. Modern techniques are limited in at least one or more the following ways: the number of neurons they can simultaneously record, the regions of the brain in which they can be deployed and in their ability to stably record from the same population of neurons over time. We briefly review each of these limitations and explain their implications.

First, modern techniques are limited in the number of neurons they can simultaneously record. Single electrodes with one recording site or those with more, such as tetrodes [16, 17], typically only record from a small handful of neural units at once. Linear arrays, which have electrodes placed at multiple locations along their shaft, can record from $10-20$ neurons at once. A single multi-electrode array (e.g, [18, 19]) typically only records from tens of neural units. Optical recording techniques (see [20] for a review) are capable of more densely sampling a region of cortex but are limited in the extent of cortex they can simultaneously image. Cutting edge technologies such as light sheet microscopy allow for the imaging of larger regions, but to date have only be demonstrated in relatively small model organisms [21]. With present day recording techniques we are therefore limited in the size of neural populations we can simultaneously monitor.

Second, current techniques for recording populations of neurons are most effective when recording from the surface of the cortex. Subcortical structures and cortical areas in sulci currently lie outside the reach of most optical methods and are typically accessed using single electrodes or linear arrays. While multiple linear arrays can be employed in principle, in practice, the placement of multiple linear arrays within a single recording session is difficult. Our ability to study populations of neurons in these areas and their interaction with populations in other areas is therefore limited.

Finally, modern electrophysiological techniques are limited in their ability to record from the same neurons across recording sessions. Chronically placed arrays are well known to pick up and drop units from day-to-day, and it is virtually impossible to record from the same set of neurons across sessions with electrodes which must be driven to their recording location every session. Scientifically, such instabilities limit our ability to apply population level analyses across days, which may be important when studying learning or when desiring to run experiments with more conditions than can be presented in a single session. Clinically, BCI systems must be recalibrated daily, which typically requires subject involvement and adds burden for end users.

1.2.1 Existing Work for Neural Stitching

This is not the first work to consider the general problem of combining neural population recordings. Indeed, there are two recent studies which also consider this problem. Importantly, neither explicitly considers structure in neural activity as a means for stitching. Turaga et al. [22], introduce the term "stitching", to describe the general problem of combining recordings of activity from two or more populations of neurons. They model neural population activity with a latent dynamical system and learn the parameters of their model using recordings from subsets of the full population of neurons modeled. Using prerecorded data from mouse sensory cortex, they show their method can predict noise correlations and couplings between neurons which were never simultaneously recorded. Soudry et al. [23] propose to use stitching to address the problem of latent common input when inferring functional connectivity between neurons. They develop a probabilistic technique for inferring functional connectivity in a stitching scenario and demonstrate the benefit that observing more neurons through stitching offers for the functional connectivity problem.

The primary point of departure of our work from this previous work is that we seek to leverage structure in neural activity, to which the studies just cited make no explicit appeal. In addition, we expand the proposed applications of stitching to BCI, the study of learning and the study of communication between neural populations in two areas of the brain. While these application areas may initially appear to have little in common, we will see that in many cases the low-dimensional structure in neural activity provides a common means of stitching. We also seek to develop a rigorous theoretical understanding of when stitching is and is not possible. This is another contribution of our work, and here again, low-dimensional structure in neural data provides a relatively straight-forward way to state the necessary conditions we will develop.

1.2.2 Thesis outline

The flow of the thesis follows the order in which I carried out my doctoral work. I began my studies in the fall of 2010 after being employed as a biomedical engineer at the Johns Hopkins University Applied Physics Laboratory (APL) on a DARPA funded neural prosthetics project.[1] The aim of the project was to develop a fully neurally integrated prosthetic limb. While there it became apparent to me that despite impressive advances in robotic technology and algorithms to decode user intent from brain signals, there was no good solution to a well known problem in intracortical BCI: the need for daily recalibration.

Removing the need to burden the patient with daily recalibration might seem to require new hardware capable of more stably recording from a population of neurons. Lacking the background to contribute to a hardware solution, I wondered if there could be an algorithmic solution. Through my work at APL I had noticed that during a typical task in which a user moves a cursor form the center of a screen to a set of peripheral targets, neural signals exhibited special structure: they clustered according to the target the user was moving the cursor to. It occurred to me that if this special structure was preserved across days, it could provide extra information an algorithm could use to recalibrate itself. As long as clusters did not move too much between days, by comparing cluster locations between two days, an algorithm could track and adjust for changes in neural signals without requiring information labelling which targets each cluster on the second day corresponded to. This provided the inspiration for the self-recalibrating classifiers presented in chapter 2. To my knowledge these were the first classifiers for intracortical BCI capable of automatic recalibration during normal device use.

The self-recalibrating classifiers were also the first example of the use of structure to combine neural recordings. In the case of the self-recalibrating classifiers, this was clustered structure that was sufficiently preserved across days. Key to the success of the self-recalibrating classifiers was a task (moves to discrete targets) which resulted in clustered neural activity.

Neural recordings made outside of BCI classification settings will not in general posses this clustered structure. I therefore turned to the development of new ML theory and algorithms to exploit this structure for combining neural population recordings. We follow Turaga et al. [22] and refer to this general idea as neural stitching. As described in chapter 4, under a common model for low-dimensional population activity, combining separate recordings of neural population activity is intimately connected to the problem of completing a partially observed covariance matrix, which is an example of a symmetric positive semidefinite (SPSD) matrix. A rigorous theoretical understanding of the completion of such matrices then provides a strong foundation to reason about conditions under which neural stitching will succeed. However, despite much work on the problem of completing general matrices when entries in the matrix are missing at random (e.g., [24–26]), it is difficult to find work that can be directly applied to the problem of completing SPSD matrices, when entries are missing in a non-random manner, as arises in the neural stitching scenario. This motivated me to study the problem of matrix completion under assumptions appropriate for neural stitching [27]. The results of this are presented in chapter 3.

With the results of chapter 3 providing the underlying theory and a basic algorithm for completion of SPSD matrices, I then turned to the problem of applying these ideas directly to the problem of combining neural population

[1] http://www.darpa.mil/program/revolutionizing-prosthetics/

recordings. This first requires formally connecting the theory developed for the problem of matrix completion to the problem of neural stitching. This was accomplished by using my matrix completion theory to develop a set of necessary conditions for fitting factor analysis (FA) models in the stitching scenario. Factor analysis allows us to infer a low-dimensional representation of an entire population of neurons, even if we are only able to observe some of the neurons in a population at any given time. In addition to using our theory to develop necessary conditions for fitting FA models in a stitching scenario, we were also able to use the algorithm presented in chapter 3 to improve the practical performance of an algorithm used to learn the parameters of FA models in the stitching scenario. These results are presented in chapter 4.

I also examined the utility of stitching in three proposed applications. In the first application, I applied stitching to the problem of developing a self-recalibrating BCI algorithm for regression. These results are presented in chapter 5. The next two applications target basic science. In the first, I examined the use of stitching to study changes in neural population activity with learning when it is impossible to record from the same set of neurons over the entire period of interest. In the second, I applied stitching to identify subspaces potentially important for communication between populations of neurons in two brain areas when it is impossible to record from large numbers of neurons simultaneously in one of the areas.

Finally, I returned to the basic problem of matrix completion. I had noticed that the sufficient conditions I developed for the completion of SPSD matrices could be stated without the need for an assumption, so called incoherence, common in much of the rest of matrix completion literature (e.g., [24, 26, 28–30]). This motivated me to examine if a well known existing technique for matrix completion, nuclear norm minimization, could be shown to work under our sufficient conditions. This is in fact the case, and the proof, as well as empirical results comparing the performance of the nuclear norm based approach to the algorithm we present in chapter 3 are presented in chapter 7.

Chapter 2

A Self-Recalibrating Classifier for Brain Computer Interface

2.1 Background

We begin this thesis by examining the problem of developing a classifier for brain-computer interface (BCI) which can recalibrate itself automatically. BCIs aim to assist disabled patients by translating recorded neural activity into control signals for assistive devices. Intracortical BCI systems rely upon arrays of chronically implanted microelectrodes which penetrate into the cortex to record the activity of tens to hundreds of neurons. The last decade has witnessed a substantial investment of resources by the community into developing decoders for intracortical BCI which are more accurate, both in offline simulations with prerecorded data and in online, closed-loop settings. However, in almost all cases these decoders are retrained daily in a supervised manner to maintain their accuracy (e.g., [4, 31–37]). This retraining typically requires that the subject perform a series of trials where the goal of each trial, such as desired reach target, is known so labelled data for training can be collected.

While retraining procedures may only require minutes of a subject's time, future users may find daily repetition of a task which stands between them and device use burdensome. Indeed, imagine the annoyance modern cell phone users would experience if they had to calibrate the touch screens of their devices everyday before use. A few minutes of calibration would be a small price to pay for the connectivity these devices provide, but it is difficult to imagine the widespread adoption of modern phones under these conditions. In the same way, we suggest that modern BCI systems should aim to achieve predictable performance each day without undue burden to users.

An alternative to retraining in a supervised manner is to employ self-recalibrating neural decoders which can simultaneously estimate a user's intent along with changes in neural tuning parameters. These decoders, generally referred to as adaptive decoders, continuously train themselves during normal BCI use without requiring knowledge of a user's true intended actions. In this way, the retraining period at the beginning of each day can be eliminated. There have been relatively few studies on self-recalibrating decoders for intracortical BCI, and the work that does exist has focused on BCI decoders which decode a continuously valued control signal, such as the position of a computer cursor or prosthetic limb [38–41].

While continuous decoders have many important applications, classifiers of discrete actions are an important part of existing real-world BCI systems [42]. Potential clinical applications for BCI classifiers include making selections from a menu interface or virtual keyboard (e.g., [32, 43, 44]), selecting among grasp types for a prosthetic hand (e.g., [45, 46]), classifying finger movements (e.g., [47, 48]), identifying periods of planning and movement (e.g., [49–52]) and providing discrete control signals for a wheelchair (e.g., [53, 54]). In some settings it may be possible to interface with a device using either a continuous or discrete control signal. For example, when interfacing with a menu, a cursor can be moved over items to be selected or a discrete decoder can be used to make a direct menu selection. In certain cases, the use of a classifier may provide a substantial performance improvement to an end user. Indeed, Santhanam et al. have demonstrated a discrete decoder which transmits the user's intent at a rate of up to 6.5 bits per second ($\sim$15 words per minute), which is faster than continuously guiding a cursor to make discrete selections [32].

Despite the potential clinical benefit that intracortical BCI classifiers may provide, the subject of self-recalibrating classifiers has received little attention in the intracortical community. In contrast, there is previous work in the EEG community which characterizes signal drift (e.g., [55, 56]), and proposes self-recalibrating classifiers. Example algorithms include approaches based on variational Kalman filtering [57, 58], expectation maximization [59] and a variety of other methods to update the parameters of an LDA classifier [60, 61]. However, many of these algorithms consider only binary classification, and it is not clear the complexity of some (e.g., [57–59]) is required for BCI application.

In the present chapter, we seek to develop self-recalibrating classifiers for intracortical BCI. We note this differs from the previous work of Santhanam et al. [62]. Santhanam et al. develop classifiers with improved within-day accuracy by accounting for unobserved influences on neural activity, while we focus on achieving stable across-day performance through self-recalibration. We chose to forgo spike sorting in favor of the use of threshold crossings. That is we mark any voltage event which crosses a set threshold on each electrode as a spike. In one monkey, we also enforce that threshold events must pass a shape heuristic. Previous work has shown only minor reductions in performance when threshold crossings are used in place of spike sorting [63, 64], and this removes the need to track and identify units across days.

Having made this design decision, we study how firing rates drift within and between recording sessions on different electrodes. This enables us to then develop self-recalibrating classifiers which take advantage of structure we find in this drift. Along the way, we also examine the performance of a standard BCI classifier that undergoes no daily retraining. In addition to being a novel result in itself, this provides a baseline when assessing the performance of the self-recalibrating classifiers presented in this chapter.

2.2 Methods

Reach task and neural recordings

We analyze prerecorded data collected in the lab of Krishna Shenoy at Stanford University. Animal protocols were approved by the Stanford University Institutional Animal Care and Use Committee. We trained two adult male monkeys (*Macaca mulatta*, monkeys I and L) to perform center-out reaches for liquid rewards. As illustrated in Fig. 2.1, visual targets were back-projected onto a vertical screen in front of the monkey. A trial began when the monkey touched a central yellow square. Following a randomized (300 - 500 ms) hold period, a visual reach target appeared. After a brief (35 ms) delay, the animal was permitted to reach. Reach targets could appear at one of twenty-eight locations. Each of the twenty-eight targets was typically located at a distance of 4.5 cm, 7.0 cm, 9.5 cm or 12 cm from the central target and at an angle of 40°, 85°, 130°, 175°, 220°, 310°, or 355° from horizontal so that the targets were arranged in 4 concentric rings of 7 targets each. For all analyses in this chapter, we collapse across reach distance and only consider the direction of reach on each trial. After the reach target was held for 200 ms for monkey L and 300 ms for monkey I, a liquid reward was delivered.

Monkeys sat in a custom chair (Crist Instruments, Hagerstown, MD) with the head braced and nonreaching arm comfortably restrained. A 96-channel silicon electrode array (Blackrock Microsystems, Salt Lake City, UT) was implanted in motor cortex contralateral to the reaching arm. When identifying neural spikes on each electrode, threshold crossings [63, 65] were used in place of spike sorting. For monkey L, we also required voltage waveforms to pass a shape heuristic. Threshold levels for each electrode were set at the beginning of each recording day with an automated method and then, with possible rare exception, held fixed for the remainder of data collection. The automated method of setting threshold levels consisted of collecting 2 seconds of filtered voltage data sampled at 30-kHz from each electrode and calculating a root-mean-square (RMS) like value from this data. Thresholds levels were then set at three times this value for each electrode.

Recordings were made on a total of 41 (monkey L) and 36 (monkey I) days spanning a period of 48 (monkey L) and 58 (monkey I) days in total. In all of the work that follows in this chapter, we limit ourselves to only analyzing successful trials. In the data analyzed, there were (mean $\pm$ s.d.) 1737 ± 482 (monkey L) and 1688 ± 368 (monkey I) successful trials per day spanning 53 ± 16 (monkey L) and 88 ± 16 (monkey I) minutes per day.

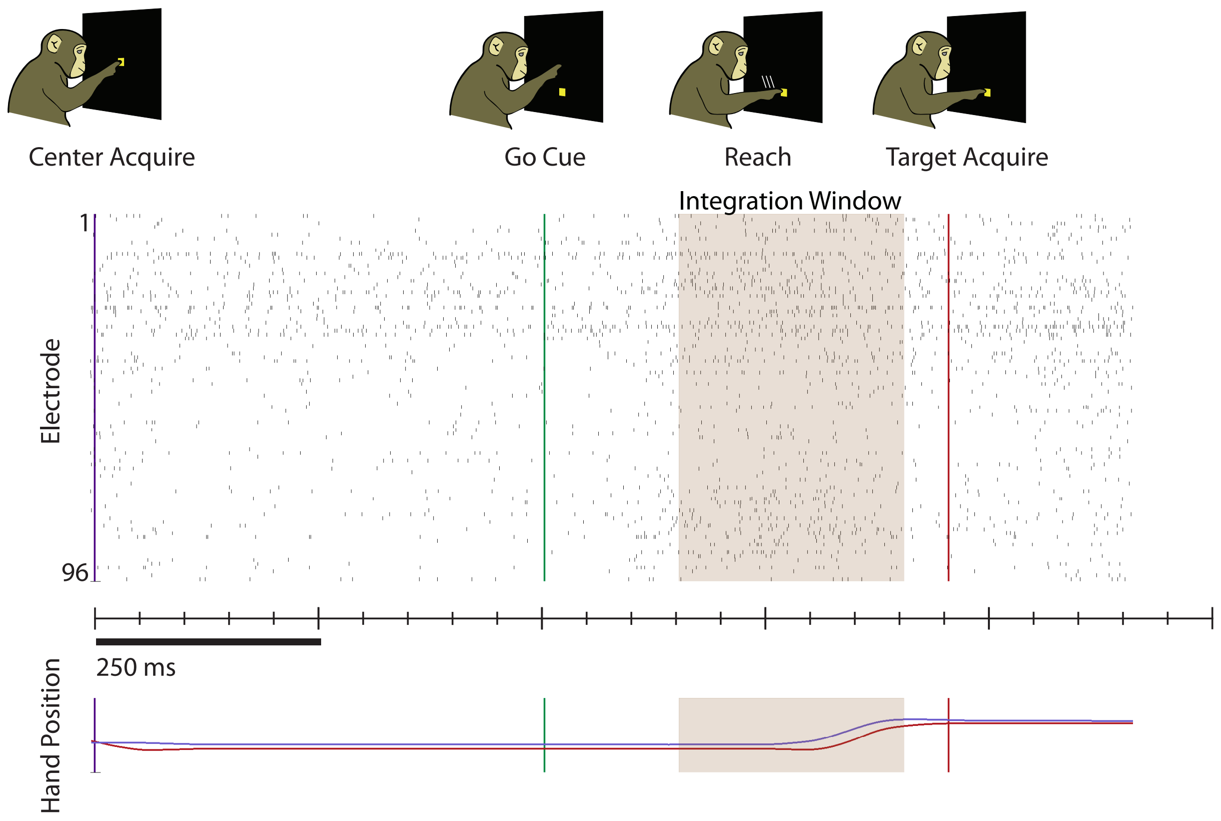


Figure 2.1: A representative trial of the center out reaching task (Monkey L, trial ID: R,2008-12-26,mm, 109). The top row is the behavioral task. The middle row contains spike rasters across 96 electrodes and the bottom row presents the timecourse of the hand position. Full range of scale for the hand position is 15 cm.

2.2.1 Mathematical Notation

We aim to develop BCI classifiers which maintain stable day-to-day performance without any daily supervised retraining. For clarity, we will speak of developing classifiers within the context of the datasets used in this study, but the techniques presented here should generalize to other intracortical BCI classification scenarios.

Throughout this chapter, we rely on probabilistic methods to infer the intended reach direction from neural data. For trial t on day d, we form a feature vector, $x_{d,t} \in \mathbb{N}^{E \times 1}$, from our neural data by counting the number of threshold crossings on each of E electrodes of the recording array. The notation $x_{d,e,t}$ will refer to the specific spike count for electrode e[1]. Similarly, $y_{d,t} \in \{1, \ldots, J\}$ indicates which one of J directions a reach is made in for a specific trial and day. The symbol P will be used to indicate probability distributions.

On any given trial there is a probabilistic model relating observed spike counts to reach direction. We model this with the distribution $P(x_{d,t}|y_{d,t}; \Theta_{d,t})$, where $\Theta_{d,t}$ is a set of parameters characterizing the distribution. In this chapter for electrode e, these parameters consist of the means $\mu_{e,j}$ and variances $v_{e,j}$ of spike counts when reaches are made in each direction j. We will refer to this set of parameters as *tuning parameters*. While both means and variances can drift in principle, we focus on achieving stable classification through tracking changes in mean spike counts. We term the set of $\mu_{e,j}$ parameters for a fixed reach direction j as class means, as they are mean counts conditioned on a class of movement. We can index class means across days and trials as $\mu_{d,e,j,t}$, where d indexes day and t indexes a trial within a day. In some of the work that follows in this chapter, we will speak of class means across all trials of a day, which will we denote as $\mu_{d,e,j}$.

[1]Throughout this chapter we will use the convention of listing subscripts in alphabetical order.

2.2.2 Tracking Drifting Tuning Parameters Across Time

Movements of the recording array relative to the brain [66], scar tissue buildup [67], behavioral changes [68] and neural plasticity [69–72] may all cause tuning parameters to drift. In this section, we track drift in class means across days and time by convolving spike counts for all reach directions in a particular direction with a Gaussian kernel. This is a simple way of smoothing spike counts to estimate tuning parameters when true reach direction is known on each trial. This is an appropriate assumption for this analysis which aims to establish basic properties of tuning parameter drift. However, the techniques introduced in this section should not be confused with the methods that are introduced later to track drifting tuning parameters by our self-recalibrating classifiers when knowledge of true reach direction is not assumed known. Mathematically, estimates of tuning parameters are formed through convolution with a kernel as

$$\hat{\mu}_{d,e,j,t} = \frac{1}{C} \sum_{s:y_{d,s}=j} K(t-s;w) x_{d,e,s}, \tag{2.1}$$

where $K(t-s;w)$ is Gaussian density function with 0 mean and standard deviation w and $C = \sum_{s:y_{d,s}=j} K(t-s;w)$ is a normalizing constant. Mathematically, the function K indexes over trial number, and therefore, $t-s$ in the argument is the distance in *number of trials* between two trials and not time. There remains the challenge of selecting the proper value of the width, w, of the Gaussian density function. For our purposes, we will present results for a single value of w and then show in appendix A that the qualitative results we obtain are robust as the value of w is varied.

We are interested in comparing the amount of drift between and within days. If we smooth only within days, we enforce smoothing within but not across days and we may bias our results to show larger between than within day drift. To account for this, we ignore day boundaries and smooth across all trials for a given electrode and reach direction as if they were on the same day. This may tend to“over-smooth” between days, but to foreshadow what is to come, we would like to argue that within-day variation in tuning parameters is small relative to between-day variation. Therefore, this procedure is conservative for our purposes.

2.2.3 A Standard BCI Classifier

We desire to examine the impact of tuning parameter drift on a standard BCI classifier. We select a Gaussian, naïve Bayes decoder as our standard classifier, which has been used for BCI classification before [32, 73]. Briefly, this classifier models spike counts on electrodes as independent Gaussian distributions conditioned on reach direction. The notation $\mu_{e,j}$ and $v_{e,j}$ refer to the mean and variance of the distribution for electrode e and reach direction j. For notational convenience, we will group all the means and variances for a single electrode e into the vectors $\mu_e \in \mathbb{R}^{J\times 1}$ and $v_e \in \mathbb{R}^{J\times 1}$, respectively. The joint probability of observing spike counts for all electrodes can be computed as the product of probabilities for individual electrodes. Formally, we write this as

$$P(x_{d,t}|y_{d,t}) = \prod_{e=1}^{E} P(x_{d,e,t}|y_{d,t}). \tag{2.2}$$

A uniform prior probability of reaching in any direction, $P(y_{d,t}) = 1/J$, is employed and Bayes' rule can then be used to find the posterior probability of reaching in direction j given neural data, $P(y_{d,t} = j|x_{d,t})$. This posterior is calculated for each reach direction, and the direction with the highest probability is selected as the decoded direction for each trial.

We will examine the performance of two versions of this standard classifier, which differ only in the way they are trained. The first version, referred to as the standard retrained classifier, learns the μ_e and v_e parameters fresh with trials from the beginning of each day on which classification will be performed. This emulates standard practice in the BCI field. The second version, referred to as the standard non-retrained classifier, is trained initially on all trials collected on a small number of days in which neural data and desired reach direction is known. After this training, the parameters of the standard non-retrained classifier are held fixed, and it is then used to predict reach direction for trials on subsequent days without any type of retraining or recalibration.

2.2.4 A Probabilistic Self-Recalibrating Classifier

We now describe the intuition and probabilistic model behind our two self-retraining classifiers. The first version, which we refer to as the SR classifier, takes a principled probabilistic approach which provides insight into the simplified self-recalibrating (SR_S) classifier we subsequently present. Readers interested only in the SR_S classifier may skip to section 2.2.5 after reading the part of this section outlining the probabilistic model common to both classifiers without loss of understanding.

The SR classifier can be considered an augmented standard classifier. It continues to assume tuning parameters are fixed within a day, but it no longer assumes class means are known at the beginning of each decoding day. Mathematically, it achieves this by introducing a distribution over class mean values that a decoding day begins with. As a decoding day progresses, the SR classifier produces refined distributions over class means while simultaneously estimating reach direction on a trial-by-trial basis.

A formal description of probabilistic model for the SR classifier follows. When explicitly referring to the class mean for electrode e, reach direction j on day d, we will use the notation $\mu_{d,e,j}$. In results we show that the $\mu_{d,e,j}$ parameters for the same electrode drift from day to day in a highly correlated manner. Intuitively, this means that the average spike counts observed for two different reach directions on the same electrode tend to move up and down together across days. We capture this intuition by referencing all class means for electrode e to the same base value $b_{d,e}$ for day d, and we allow this base value to be drawn from a Gaussian distribution independently on each day. Mathematically,

$$b_{d,e} \sim \mathcal{N}\left(m_e,\ s_e\right), \tag{2.3}$$

where m_e is the mean and s_e is the variance of the distribution from which base values are drawn. For notational convenience we will group all the base values for all electrodes for day d into a vector $b_d \in \mathbb{R}^{E\times 1}$. We model the base values for different electrodes as independent, and the prior probability for the set of base values for all electrodes is therefore given by

$$P(b_d) = \prod_{e=1}^{E} P(b_{d,e}). \tag{2.4}$$

We relate the class mean for reach direction j for an electrode to its base value through an offset, $o_{e,j}$, so that

$$\mu_{d,e,j} = o_{e,j} + b_{d,e}. \tag{2.5}$$

The offset, $o_{e,j}$, is fixed across days. Class means for a single electrode can then vary from day to day but they must move up and down together. Having defined a probabilistic model for $b_{d,e}$, we specify an observation model for a single electrode by defining the probability of a spike count on electrode e for trial t on day d given $b_{d,e}$ and reach direction j as

$$x_{d,e,t} | b_{d,e}, y_{d,t} = j\ \sim\ \mathcal{N}\left(o_{e,j} + b_{d,e}, v_{e,j}\right). \tag{2.6}$$

This observation model is almost identical to that of the standard classifier with one important exception: spike counts now depend on the base value, which is a random variable that can vary from day to day. To specify the complete observation model, we again model electrodes as conditionally independent given reaching direction so that

$$P(x_{d,t}|b_d, y_{d,t}) = \prod_{e=1}^{E} P(x_{d,e,t}|b_{d,e}, y_{d,t}). \tag{2.7}$$

To complete the specification of the probabilistic model for the self-recalibrating classifier, we define the prior probability of reaching in any direction as uniform, exactly as specified for the standard model.

Decoding with the Self-Recalibrating Classifier

We now show how to decode reach direction while simultaneously refining knowledge of class means with the SR classifier. For each trial, we seek to find two distributions. The first is $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$, the posterior distribution over reach directions. The most likely direction from this distribution is selected as the decoded reach direction on each trial. The second distribution is $P(b_d|\{x_{d,s}\}_{s=1}^{t})$, which embodies the decoder's knowledge of the base values for a day given the neural activity that it has seen.

As a decoding day progresses, the decoder's knowledge of the true base values for a day, as represented in the distribution $P(b_d|\{x_{d,s}\}_{s=1}^{t})$, becomes more refined. This is possible even though the decoder is never provided with knowledge of the user's true reach directions. The key intuition is that spike counts will tend to cluster around class means for a day, and this structure can be exploited by our classifier, as illustrated for a one electrode and two reach direction decoding scenario in Fig. 2.2.

We now describe the formal decoding procedure for arriving at the distributions $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ and $P(b_d|\{x_{d,s}\}_{s=1}^{t})$ for each trial. The algorithm is recursive and will use the results from the previous trial, the distributions $P(y_{d,t-1}|\{x_{d,s}\}_{s=1}^{t-1})$ and $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$, as its starting point in its calculations for trial t. When decoding the first trial, the prior distribution on reach directions, $P(y_{d,t})$, is used in place of $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t-1})$ and the prior distribution on base values, $P(b_d)$, is used in place of $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$.

Calculating $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$. For trial t, we calculate $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ using Bayes' rule as

$$\begin{aligned} P(y_{d,t}=j|\{x_{d,s}\}_{s=1}^{t}) &= \frac{P(x_{d,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)P(y_{d,t}=j|\{x_{d,s}\}_{s=1}^{t-1})}{P(x_{d,t}|\{x_{d,s}\}_{s=1}^{t-1})} \\ &= \frac{c_{t|j}P(y_{d,t})}{c_t}, \end{aligned} \tag{2.8}$$

where in moving to the second line we have defined $c_{t|j} = P(x_{d,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)$, made use of the fact that reach direction is independent of neural activity so that $P(y_{d,t}=j|\{x_{d,s}\}_{s=1}^{t-1}) = P(y_{d,t})$, and recognized that the denominator of the first line is a normalizing constant calculated as $c_t = \sum_{j=1}^{J} c_{t|j}P(y_{d,t})$.

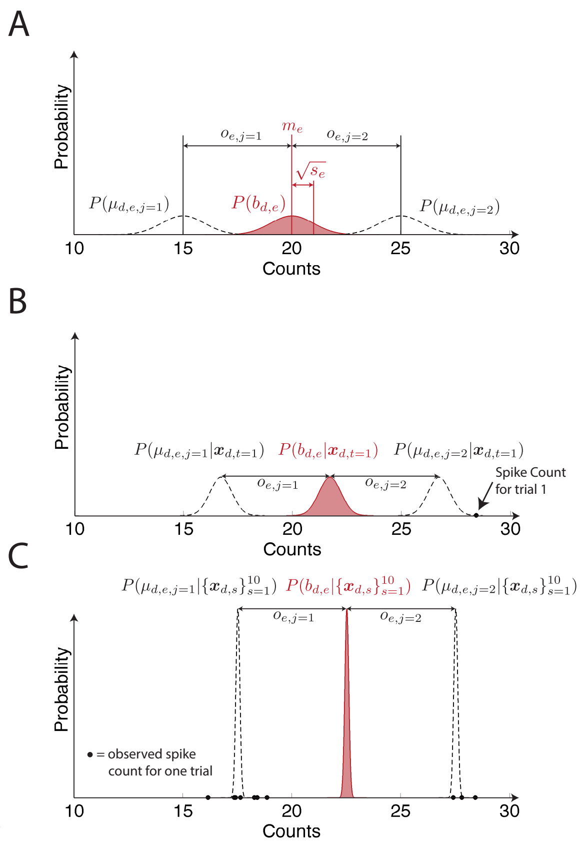


Figure 2.2: An illustration of the probabilistic distribution over tuning parameters for the self-recalibrating classifier at three different points in time during a decoding session on day d for an example where spike counts recorded on one electrode are used to predict reaches in one of two directions. (A) The distribution over the base value for the day, $b_{d,e}$, for the electrode at the start of the decoding day before any neural activity is available. This distribution is a Gaussian distribution with mean m_e and variance s_e. In all panels, distributions over $b_{d,e}$ are shown in red to indicate that this is the distribution the self-recalibrating classifier explicitly represents. Panel A also shows the distributions over $\mu_{d,e,j=1}$ and $\mu_{d,e,j=2}$, which are the mean number of spike counts conditioned on reaching in directions 1 and 2. The classifier does not explicitly represent these distributions, but they can be recovered from the distribution over $b_{d,e}$ with the formula $\mu_{d,e,j} = b_{d,e} + o_{e,j}$. (B) The distributions over $b_{d,e}$, $\mu_{d,e,j=1}$ and $\mu_{d,e,j=2}$ after spike counts for one trial have been observed. Importantly, note that the true reach direction associated with this spike count is not made available to the classifier. (C) Distributions over the same quantities after 10 trials have been observed. The solid dots in panels B and C represent spike counts observed on individual trials.

The scalar $c_{t|j}$ is the probability of the neural activity for trial t given all past neural activity seen during the day if the subject reaches in direction j. We will soon describe a means of calculating its value, but first note that it appears in the calculations for another distribution, $P(b_d|\{x_{d,s}\}_{s=1}^{t}, y_{d,t} = j)$, which we will need shortly. This distribution gives the probability of base values conditioned on all observed neural activity if the subject were to reach in direction j for the current trial. To calculate it, we again use Bayes' rule to write

$$P(b_d|\{x_{d,s}\}_{s=1}^{t}, y_{d,t} = j) = \frac{P(x_{d,t}|b_d, y_{d,t} = j)P(b_d|\{x_{d,s}\}_{s=1}^{t-1})}{c_{t|j}}, \tag{2.9}$$

where $P(x_{d,t}|b_d, y_{d,t} = j)$ is given by eq. 2.7 and $P(b_d|\{x_s\}_{s=1}^{t-1})$ is the posterior over base values obtained when decoding the previous trial. In forming the numerator of eq. 2.9, we have used the fact that $x_{d,t}$ conditioned on b_d is independent of all previous neural observations and that base value is independent of reach direction. We see that the numerator on the right hand side of eq. 2.9 is in the form of the product of two Gaussians. $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$ is defined to be a Gaussian distribution with mean m_t and covariance matrix Σ_t. Viewed as a function of b_d, $P(x_{d,t}|b_d, y_{d,t} = j)$ is Gaussian with mean $x_{d,t} - o_j$, where $o_j \in \mathbb{R}^{E\times 1}$ is a vector of offsets for each electrode for reach direction j and covariance matrix $V_j \in \mathbb{R}^{E\times E}$, where V_j is a diagonal matrix with the values of $v_{e,j}$ along it's diagonal. We can then calculate the mean, $m_{t|j}$, and covariance matrix, $\Sigma_{t|j}$, of $P(b_d|\{x_{d,s}\}_{s=1}^{t}, y_{d,t} = j)$ of the left hand side of eq. 2.7 as

$$\Sigma_{t|j} = \left(V_j^{-1} + \Sigma_t^{-1}\right)^{-1} \tag{2.10}$$

$$m_{t|j} = \Sigma_{t|j}\left(V_j^{-1}(x_{d,t} - o_j) + \Sigma_t^{-1} m_t\right), \tag{2.11}$$

where the notation $\cdot^{-1}$ indicates matrix inversion. The scalar $c_{t|j}$ is then the integral over the numerator of eq. 2.9 and can be calculates as

$$\begin{aligned} c_{t|j} &= \int P(x_{d,t}|b_d, y_{d,t} = j)P(b_d|\{x_{d,s}\}_{s=1}^{t-1})dx_{d,t} \\ &= \sqrt{(2\pi)^E \det\left(\Sigma_{t|j}\right)}P(x_{d,t}|b_d = m_{t|j}, y_{t,d} = j)P(b_d = m_{t|j}|\{x_{d,s}\}_{s=1}^{t-1}), \end{aligned} \tag{2.12}$$

where $P(x_{d,t}|b_d = m_{t|j}, y_{t,d} = j)$ is obtained by evaluating eq. 2.7 for the current observation at $b_d = m_{t|j}$ and $P(b_d = m_{t|j}|\{x_{d,s}\}_{s=1}^{t-1})$ is obtained by evaluating the posterior over base values from the last trial at the same point, $b_d = m_{t|j}$.

Calculating $P(b_d|\{x_s\}_{s=1}^{t})$. Finally, we compute $P(b_d|\{x_s\}_{s=1}^{t})$, the posterior distribution over the base values for all electrodes. An analytical expression for this can be written as

$$P(b_d|\{x_{d,s}\}_{s=1}^{t}) = \sum_{j=1}^{J} P(b_d|\{x_{d,s}\}_{s=1}^{t}, y_{d,t} = j)P(y_{d,t} = j|\{x_{d,s}\}_{s=1}^{t}), \tag{2.13}$$

where the two terms on the right hand side of eq. 2.13 are obtained from eqs. 2.8 and 2.9. The right hand side of equation eq. 2.13 is a mixture of multivariate Gaussians. This will be problematic as in general there will be an exponentially increasing number of components in the representation of $P(b_d|\{x_{d,s}\}_{s=1}^{t})$ as t increases. Therefore, we approximate the right hand side of eq. 2.13 with mean m_t and covariance matrix Σ_t using moment matching as has been done in previous hypothesis merging algorithms [74]. We use the standard formulas for moment matching with two multivariate Gaussians to write

$$m_t = \sum_{j=1}^{J} m_{t|j} P(y_{d,t} = j|\{x_{d,s}\}_{s=1}^{t}) \tag{2.14}$$

$$\Sigma_t = \sum_{j=1}^{J} \left[\Sigma_{t|j} + (m_{t|j} - m_t)(m_{t|j} - m_t)^T\right] P(y_{d,t} = j|\{x_{d,s}\}_{s=1}^{t}), \tag{2.15}$$

where $P(y_{d,t} = j|\{x_{d,s}\}_{s=1}^{t})$ is calculated from eq. 2.8. This completes the decoding procedure for trial t. For each new trial, this set of calculations is repeated in an iterative manner.

Learning the Parameters for the Self-Recalibrating Classifier

While the self-recalibrating classifier is designed to produce stable day-to-day classification without daily retraining, it must be initially trained once. To perform this initial training there must exist a small number of days for which neural activity, $x_{d,t}$, has been recorded along with associated reach directions, $y_{d,t}$. We will use the notation $\mathcal{D}_{\text{Train}}$ to refer to this set of days. To avoid any possible confusion, we stress that after the classifier has been trained on this initial set of days, it requires no further supervised retraining and will be equipped to autonomously calibrate itself on new data that it encounters during normal decoding use.

The purpose of the initial training is to learn the mean, m_e, and variance, s_e, characterizing the distribution of base values for each electrode from eq. 2.3 and the offset, $o_{e,j}$, and variance, $v_{e,j}$, characterizing the observation model in eq. 2.6 for each electrode-class pair. Fitting the self-recalibrating classifier is more difficult than fitting the standard classifier because the daily base values for each electrode, $b_{d,e}$, cannot be directly observed. Instead, these values are randomly drawn each day from the distribution specified by eq. 2.3, and after the base values are drawn on each day, they specify the mean of the Gaussian distributions in eq. 2.6 from which neural spike counts are drawn. Thus, the neural data we directly observe is essentially one step removed from the distribution that is characterized by the parameters we desire to estimate. Fig. 2.3 illustrates the conceptual difference between the standard and self-recalibrating classifier models.

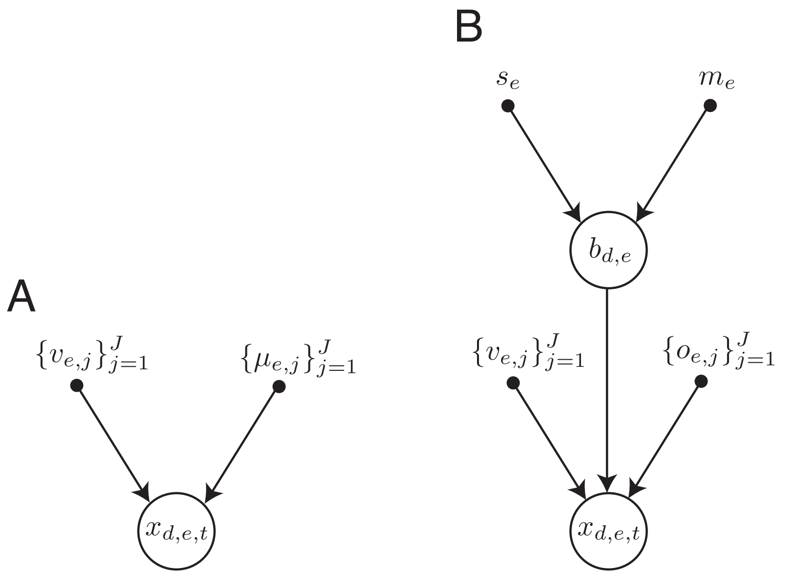


Figure 2.3: Graphical models for the generation of a single spike count on electrode e for trial t on day d for the standard (A) and self-recalibrating (B) classifiers. Circles denote random variables and dots indicate model parameters. In the standard classifier, the electrode-class means, $\mu_{e,j}$, and variances, $v_{e,j}$, for all reach directions directly characterize the distribution over observed spike counts, $x_{d,e,t}$. For the self-recalibrating classifier, the base value for electrode e, $b_{d,e}$, is randomly generated each day from a Gaussian distribution with mean m_e and variance s_e. For a given base value, spike counts are produced according to a distribution with electrode-class means $\mu_{e,j} = b_{d,e} + o_{e,j}$ and variances $v_{e,j}$. In this figure, the notation $\{v_{e,j}\}_{j=1}^J$ and $\{\mu_{e,j}\}_{j=1}^J$ indicates the set of variances and means, respectively, for all reach directions for electrode e.

The expectation-maximization (EM) algorithm [75] is a general procedure for finding the maximum likelihood model parameters in the presence of unobserved variables, in this case $b_{d,e}$. The EM algorithm is an iterative procedure which is initialized with an initial guess for the model parameters. In this case, these are the m_e, s_e, $o_{e,j}$ and $v_{e,j}$ parameters. Each iteration of the algorithm produces new estimates for these parameters through a two-step procedure. In the first step (referred to as the E-step) of each iteration, the latest parameter estimates and observed data are used to form a posterior distribution over the unobserved variables conditioned on the observed data. For the self-recalibrating classifier, the unobserved variables are the daily base values for each electrode, $b_{d,e}$, and the observed data are the neural observations, $x_{d,t}$, and reach directions, $y_{d,t}$, for all trials on all days in the training set. In the second step of each iteration (referred to as the M-step), parameter values are updated so that the expectation of the log-likelihood of the observed and unobserved data, computed using the posterior distribution found in the first step of each iteration, is maximized. This algorithm is known to converge to a local optimum. See appendix A for specific details of the EM algorithm.

Robust Classification Through Outlier Detection

The classification procedure described above can be modified to improve robustness on real-world data. Key to the probabilistic model underlying the self-recalibrating classifier is the specification that firing rates can change between but not within days. While minor changes in firing rate throughout a day are tolerable, large changes, which may be observed in real data, can be problematic.

In particular, the self-recalibrating classifier represents knowledge of the base values for a day obtained from classifying trials 1 through t with the distribution $P(b_d|\{x_{d,s}\}_{s=1}^{t})$. This distribution is refined for each trial that is classified, and over time the uncertainty for the base values for a day, represented in the covariance matrix Σ_t, is reduced. This presents no problem in normal operation. However, if the firing rate of an electrode suddenly changes after the uncertainty in base values for a day is reduced, spike counts recorded on that electrode will strongly and erroneously affect classification.

The classification procedure can be modified by adding an additional step to be performed before any of the steps already described when classifying reach direction for trial t. In this step, electrodes with large firing rate changes are flagged, and the uncertainty in the estimate for the base value for those electrodes is increased to a predetermined value. This prevents those electrodes from dominating when determining reach direction probability and maintains classification accuracy. As classification continues, uncertainty around the base value estimates for the flagged electrodes is reduced starting from the predetermined value as the self-recalibrating classifier refines the distribution $P(b_d|\{x_{d,s}\}_{s=1}^{t})$.

To flag erratically behaving electrodes, a distribution which predicts the probability of the spike count on trial t for electrode e given all previous neural data, $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1})$, is formed for each electrode as shown in section A.1.2. This can be used as a gauge for determining if the observed spike count, $x_{d,e,t}$, for a trial and an electrode is anomalous by setting upper and lower thresholds on the value of $x_{d,e,t}$ so that a normally behaving electrode will only exceed these values approximately 1% of the time. Whenever an electrode does exceed these values, it is flagged as behaving erratically. As with any anomaly detection procedure, thresholds have to be set to balance true and false positive rates, and a false positive rate of 1% should not significantly affect classification performance.

After one or more electrodes are flagged as behaving erratically for trial t, uncertainty around the estimates for their associated base values in the distribution $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$ are increased. For ease of notation, we will momentarily switch from representing the set of base values in vector notation to set notation, so that each entry in the vector b_d will be represented by a corresponding member in the set $\{b_{d,e}\}_{e=1}^{E}$. We will refer to the set of flagged electrodes for trial t as $E_{\text{t,flagged}}$ and the set of non-flagged electrodes as $E_{\text{t,non-flagged}}$. The uncertainty for the base values of erratically behaving electrodes represented in the distribution for $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ is reset in a two step procedure. In the first step, the distribution over only non-flagged electrodes, $P(\{b_{d,e}\}_{e\in E_{\text{t,non-flagged}}}|\{x_{d,s}\}_{s=1}^{t-1})$, is formed. Formally, this is accomplished by forming the marginal distribution over the electrodes in $E_{\text{t,non-flagged}}$ from $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ and is easily accomplished since $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ is a Gaussian distribution. In the second step, a new distribution over the base values for all electrodes, which we refer to as $P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$, is formed as

$$P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1}) = P(\{b_{d,e}\}_{e\in E_{\text{t,non-flagged}}}|\{x_{d,s}\}_{s=1}^{t-1}) \prod_{e\in E_{\text{t,flagged}}} P'(b_{d,e}), \tag{2.16}$$

where $P'(b_{d,e})$ are Gaussian distributions. The mean of the distribution $P'(b_{d,e})$ is set to, $m_{e,t-1}$, which is the element for electrode e of the mean vector for the distribution $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$. This ensures that the mean of the final distribution, $P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$, is the same as that of original distribution $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$. The

variance of the distribution $P'(b_{d,e})$ is set to s_e, the variance of the distribution that base values are modeled as being drawn from at the start of each decoding day. Computationally, these two steps can be carried out through a series of simple manipulations of the covariance matrix for $P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ and increase the uncertainty around base values for suspect electrodes in $P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$. After the distribution for $P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ is formed, classification, as described by the algorithm above, is performed where
$P(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ is replaced with $P^*(\{b_{d,e}\}_{e=1}^{E}|\{x_{d,s}\}_{s=1}^{t-1})$ as a starting point when decoding trial t.

2.2.5 A Simplified Self-Recalibrating Classifier

The SR classifier addresses the problem of base values, b_d, which can drift from day to day by starting with a prior distribution, $P(b_d)$, over these values and then forming refined distributions, $P(b_d|\{x_{d,s}\}_{s=1}^{t})$, over base values based on neural activity seen throughout the decoding day. When predicting reach direction for a trial, the SR classifier then considers the probability of reach directions under all possibly values for b_d and weights its estimate according to how likely each value of b_d is, given by $P(b_d|\{x_{d,s}\}_{s=1}^{t})$.

Thus, the SR classifier never assumes any single value of b_d is correct, but decodes by considering an infinite number b_d values, each weighted by some probability. This represent a fully probabilistic means of decoding. However, it is not the only possible means of accounting for drifting tuning parameters. In particular, we can forego modeling uncertainty in b_d and instead decode with a single point estimate for b_d formed for each trial. For the simplified classifier, b_d is defined so that $b_{d,e}$ is the average number of spike counts on electrode e for any trial on day d. We can express this in mathematical notation as $b_{d,e} = \mathbb{E}\left[x_{d,e,t}\right]$.

An estimate for $b_{d,e}$ on trial t, which we refer to as $\hat{b}_{d,e,t}$, can be formed by taking the empirical average of all of the observed spike counts for electrode e on day d up to and including trial t. This estimate can be recursively formed as

$$n_{d,t} = n_{d,t-1} + 1, \tag{2.17}$$

$$\hat{b}_{d,e,t} = \frac{n_{d,t-1}\hat{b}_{d,e,t-1} + x_{d,e,t}}{n_{d,t}}. \tag{2.18}$$

We must select initial values for $n_{d,0}$ and $b_{d,e,0}$, where the hat has been dropped on $b_{d,e,0}$ to indicate this is a parameter of the model specifying a value with which to initialize eq. 2.18. If we select $n_{d,0} = 0$, $\hat{b}_{d,e,t}$ will be the true empirical average for trials 1 through t. However, this has one disadvantage. Because of the very fact that eq. 2.18 forms estimates as the empirical average of spike counts, at the start of the day there will be no previous trials with which to average spike counts for initial trials with and hence values for $\hat{b}_{d,e,t}$ will be strongly influenced by the spike counts of each trial. This will result in large fluctuations in $\hat{b}_{d,e,t}$ for the early trials of a decoding session. An alternative is to select $b_{d,e,0}$ so that it represents the expected average spike count on electrode e for any given day and assign this some initial weight by setting $n_{d,0} > 0$. Intuitively, we can think of starting each day with an initial estimate of the average spike count for the electrode that was obtained as the empirical average of $n_{d,0}$ "virtual samples." This initial estimate is analogous to the prior distribution for the fully probabilistic classifier specified in eq. 2.3. Importantly, as shown in appendix A, as long as $n_{d,0}$ and $b_{d,e,0}$ are chosen to be finite, $\hat{b}_{d,e,t}$ approaches the true mean firing spike count for electrode e as the number of trials observed during a day grows. While this is reassuring, there are still values of these parameters that will result in optimal decoding, and below we discuss how values for $n_{d,0}$ and $b_{d,e,0}$ can be learned from the training data.

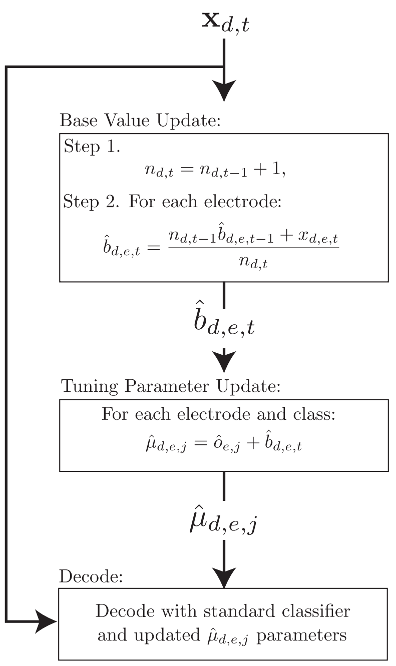

$\mathbf{x}_{d,t}$
Base Value Update:
Step 1.
$n_{d,t} = n_{d,t-1} + 1,$
Step 2. For each electrode:
$\hat{b}_{d,e,t} = \frac{n_{d,t-1}\hat{b}_{d,e,t-1} + x_{d,e,t}}{n_{d,t}}$
$\hat{b}_{d,e,t}$
Tuning Parameter Update:
For each electrode and class:
$\hat{\mu}_{d,e,j} = \hat{o}_{e,j} + \hat{b}_{d,e,t}$
$\hat{\mu}_{d,e,j}$
Decode:
Decode with standard classifier and updated $\hat{\mu}_{d,e,j}$ parameters

Figure 2.4: An illustration of how the SR_S classifier predicts reach direction for a single trial. Starting from the top, spike counts, $x_{d,t}$, for trial t are recorded and the SR_S decoder updates its estimate for the day's base values, $\hat{b}_{d,e,t}$, using a simple accumulated average. Updated estimates for tuning parameters, $\hat{\mu}_{d,e,j}$, are then calculated from the updated estimates of the base values. Finally, the standard classifier is used with the updated tuning parameters to predict reach direction.

After an estimate for $\hat{b}_{d,e,t}$ is formed for each trial, an estimate for the class means, $\mu_{d,e,j}$, can be obtained with eq. 2.5, and a classification of reach direction can be made using the standard classifier described in section 2.2.3 with the updated class means. The SR_S classifier can be conceptualized as consisting of two algorithms acting together. The first algorithm tracks base values, and hence tuning parameters, using an accumulated average of spike counts. This simple first algorithm is independent of and provides input in the form of updated tuning parameters to the second algorithm, which serves as the decoding core and is nothing more than the standard classifier. Fig. 2.4 presents a schematic depiction of how the SR_S classifier decodes a single trial and depicts the separation of the self-recalibrating and classification portions of the SR_S decoding algorithm. See appendix A for a more formal description of the relationship between the SR and SR_S classifiers. We note this method of tracking baseline firing rates is similar to the work of Li et al. [41], who in their work on adaptive decoders for continuous control signals experimented with tracking changes in baseline firing rates by measuring the empirical average of spike counts. However, since their algorithm runs in a batch mode and does not update parameters on a single trial basis as our does, they do not introduce the concept of selecting initial values and weights to begin the empirical average for a day with.

Learning Model Parameters for the Simplified Self-Recalibrating Classifier

As with the SR classifier, the SR_S classifier must also be trained once on an initial set of training days. The goal of this training is to learn the electrode-class variances, $v_{e,j}$, and the electrode-class offsets, $o_{e,j}$, for all electrodes and reach directions. In addition, values for $b_{d,e,0}$, must be learned for each electrode, and the initial weight, n_0, which will be assigned to these means must be determined.

Given a collection of days, $\mathcal{D}_{\text{Train}}$, we estimate values for $b_{d,e,0}$, $o_{e,j}$ and $v_{e,j}$ as follows. First, for each day in $\mathcal{D}_{\text{Train}}$ we estimate the average spike counts on each electrode over all reach directions, $\hat{\mu}_{d,e}$, and the average spike counts on each electrode when the subject made reaches in direction j, $\hat{\mu}_{d,e,j}$. We calculate

$$\hat{\mu}_{d,e} = \frac{\sum_{t=1}^{T_d} x_{d,e,t}}{T_d}, \tag{2.19}$$

and

$$\hat{\mu}_{d,e,j} = \frac{\sum_{t:y_{d,t}=j} x_{d,e,t}}{T_{d,j}}, \tag{2.20}$$

where T_d is the number of trials on day d, and $T_{d,j}$ is the total number of trials with reaches in direction j on day d. Having calculated these intermediate values, an estimate of $b_{d,e,0}$ is then calculated as the average of the daily average spike counts as

$$\hat{b}_{d,e,0} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} \hat{\mu}_{d,e}}{|\mathcal{D}_{\text{Train}}|}, \tag{2.21}$$

where $|\mathcal{D}_{\text{Train}}|$ is the number of days in $\mathcal{D}_{\text{Train}}$. Estimates for the class-electrode offsets, $o_{e,j}$, are calculated as the average of the difference between daily class-electrode mean spike counts and the daily average spike counts. We write this as

$$\hat{o}_{e,j} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} \hat{\mu}_{d,e,j}}{|\mathcal{D}_{\text{Train}}|} - \hat{\mu}_{d,e}. \tag{2.22}$$

Finally, the electrode-class variances, $v_{e,j}$, are estimated as

$$\hat{v}_{e,j} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} \sum_{t:y_{d,t}=j} (x_{d,e,t} - \hat{\mu}_{d,e,j})^2}{\left(\sum_{d \in \mathcal{D}_{\text{Train}}} T_{d,j}\right) - 1}, \tag{2.23}$$

which can be understood as a generalization of the standard formula for the variance of a sample generalized to account for values of $\mu_{d,e,j}$ which may change between days.

Learning the value of n_0 The equations above are used to estimate values for the $b_{d,e,0}$, $o_{e,j}$ and $v_{e,j}$ parameters of the model. We turn to cross-validation to learn a value for n_0, which is the amount of weight or number of virtual samples that should be assigned to the initial estimate of marginal spike counts for a day, $b_{d,e,0}$.

In the cross-validation procedure, we seek the value of n_0 which will produce the highest classification accuracy on novel data. To do this, multiple rounds of training and validation are performed on the data that is present in $\mathcal{D}_{\text{Train}}$. In each round, one day in $\mathcal{D}_{\text{Train}}$ is set aside as a validation day. On the remaining days, the $b_{d,e,0}$, $o_{e,j}$ and $v_{e,j}$ parameters of the model are learned. Importantly, these values are learned on all days in $\mathcal{D}_{\text{Train}}$ except the validation day. This is critical because cross-validation uses the validation day to simulate classification on novel data, and it is therefore important that none of the trials in the validation day are used for learning any part of the model for a round of cross-validation. After learning these parameters, a value of n_0 is selected and the classifier is used to predict reach direction for each trial in the validation day. This is repeated for a range of n_0 values. For a single round of cross-validation, one or more values of n_0 will produce the highest classification accuracy on the validation day. To select the final value of n_0 for use in the trained classifier, a round of cross validation is performed for each day in $\mathcal{D}_{\text{Train}}$, and the value of n_0 which produces the highest average accuracy across all rounds is chosen.

2.3 Results

2.3.1 Tracking Tuning Parameter Drift Across Time

Figs. 2.5 and 2.6 show the results of the analysis of drifting tuning parameters for monkeys L and I, respectively. Throughout this chapter threshold crossings were collected in a single window per trial. In this analysis and in all of the work in the rest of this chapter, we count spikes falling in a single 250 millisecond long window for each trial starting 150 milliseconds after the go cue. As described in section 2.2.2, spike counts from each electrode were then convolved with a Gaussian kernel to produce smoothed estimates of tuning parameters across trials. The results presented here were obtained with a kernel width of 100 trials. Results obtained with kernel widths of 25, 50 and 200 trials were qualitatively similar (see appendix A).

Panel A in Fig. 2.5 shows the traces of two tuning parameters for one electrode for monkey L. Traces for monkey I were similar but are not shown. Three important properties of tuning parameter drift are apparent in this plot. First, tuning parameters change much more between than within days. Second, tuning parameters for the same electrode tend to move up and down together across days. Finally, tuning parameters do not tend to drift progressively further from their values on day 1 over time.

Fig. 2.5B and Fig. 2.6A provide a quantitative measurement of the observation that tuning parameters appear to drift more between than within days. The drift of a tuning parameter within a single day is quantified as the standard deviation of its estimated values over all of the trials within a day, $\sigma_{d,e,j}$. This can be measured for all days, electrodes and reach directions. The distribution of all $\sigma_{d,e,j}$ values is plotted as the dashed line in each figure. Similarly, drift between days is quantified as the standard deviation of mean tuning parameter values for each day, $\sigma_{e,j}$. The distribution of $\sigma_{e,j}$ parameters is plotted as the solid line in both figures. The mean of the between-day standard deviation values is significantly larger than that of the within-day standard deviation values (permutation test, $p < .01$). This finding motivates the decision to simplify the self-recalibrating classifiers so that they ignore within-day drift but compensate for between-day drift in tuning parameters.

Fig. 2.5D and Fig. 2.6B provide evidence that tuning parameters on the same electrode drift in a correlated manner. Correlation coefficients for all pairs of daily mean tuning parameters (the black dots for the example pair of tuning parameters in Fig. 2.5C) on the same electrode were calculated for each electrode. We also examined correlation across electrodes by measuring the correlation of daily mean values for pairs of tuning parameters for the same reach direction across all pairs of electrodes. The distribution of these two sets of correlation values is shown in Fig. 2.5D and Fig. 2.6B. For both subjects, correlation values for tuning parameters on the same electrode concentrate near 1. This insight has important practical implications for the design of self-recalibrating classifiers which will be detailed in section 2.4.

While tuning parameters drift in a correlated manner on the same electrode, it is important to point out that the distributions of correlation coefficients in Fig. 2.5D and Fig. 2.6B leave room for tuning parameters to move in a limited manner relative to one another while still moving up and down together. Indeed, this is visible in Fig. 2.5A where traces of the two tuning parameters move up and down together between days but are closer to one another on days 1-4 than they are for days 7-10, for example. We will show below that our decoders, which use a simplified design that models parameters on the same electrode as moving in perfect lockstep, achieve good performance, but future decoders could be developed to model additional aspects of tuning parameter drift. Interested readers may refer to appendix A, where we quantify changes in modulation depth across time.

Fig. 2.5E and Fig. 2.6C measure the tendency of tuning parameters to progressively drift from their value on day 1. The percent change of the mean tuning parameter values from day 1, $\Delta_{d,e,j}$, for days 2 and on for all tuning parameters was calculated. The average of the absolute value of the percent change in tuning parameter values across all electrodes and reach directions for each day was then plotted in these figures. When performing this analysis, for numerical reasons we removed any electrodes which had a value of less than .001 spikes/window for any reach direction on day 1. The results are shown in Fig. 2.5E and Fig. 2.6C. If tuning parameters drift in an unconstrained manner, the average of the unsigned $\Delta_{d,e,j}$ values should monotonically increase with time. However, average values do not appear to systematically increase with time, supporting the conclusion that tuning parameters can drift from day to day but do so in a constrained manner.

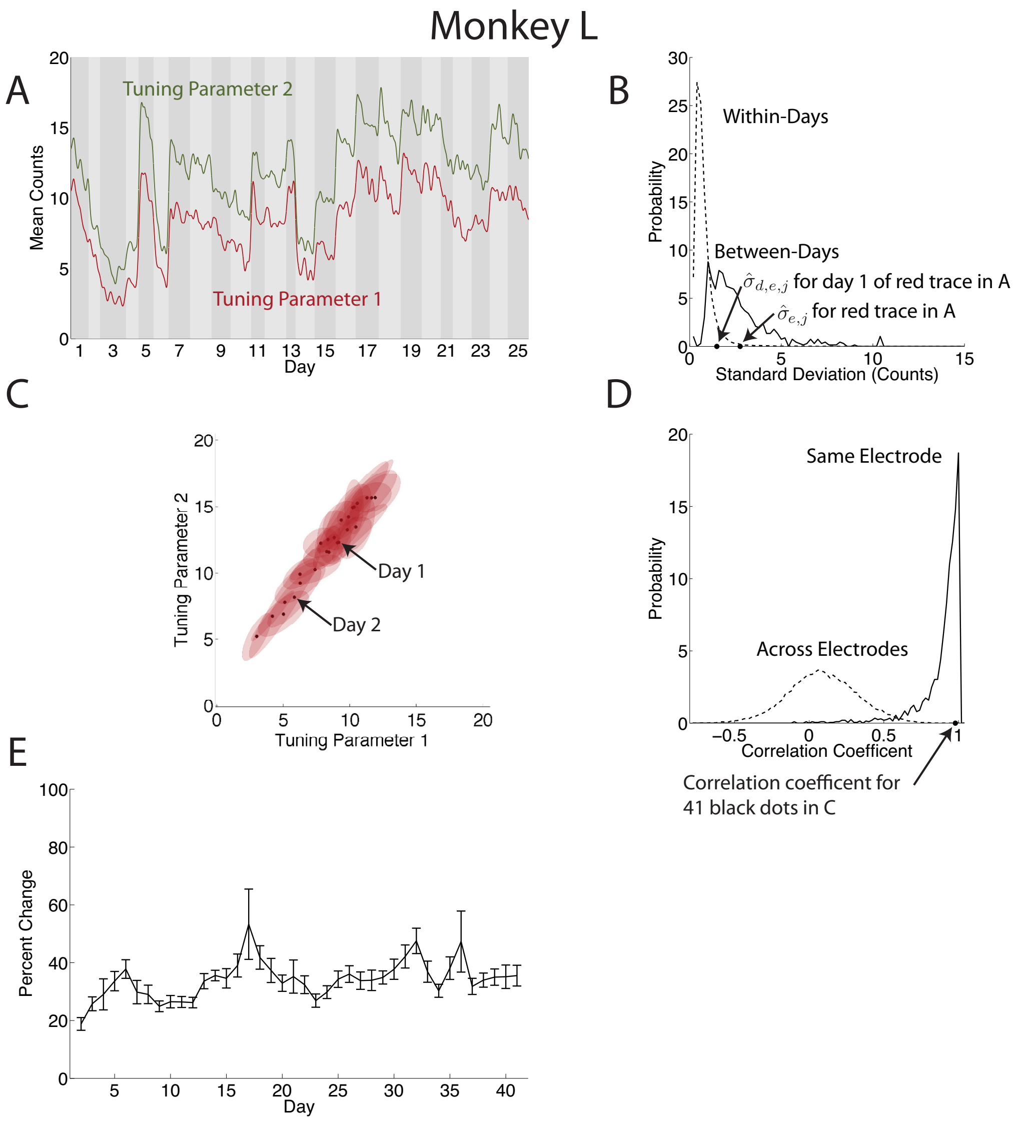


Figure 2.5: Statistical structure of changes in tuning parameters across days for monkey L. (A) Example traces of $\hat{\mu}_{d,e,j,t}$ for two representative tuning parameters on the same electrode across the first 25 days of data. Background shading represents day boundaries. (B) Distribution of the between ($\sigma_{e,j}$ for all e and j) and within-day ($\sigma_{d,e,j}$ for all d, e, and j) standard deviations of tuning parameters. A small number $(.08\%)$ of within-day standard deviations are outside the range of this plot. (C) Daily means (black dots) and associated ellipses which indicate regions in which tuning parameters for each day fall with approximately 95% probability for the same $\hat{\mu}_{d,e,j,t}$ parameters shown in panel A on each of the 41 days of recorded data. (D) Distribution of correlation coefficients computed across and within electrode pairs. To relate panel C to D, the correlation of the cloud of 41 black points in panel C is *one* of the $96 \times \binom{7}{2}$ pairs of correlations that was used to construct the "same electrode" distribution in panel D. (E) Average values with 95% confidence intervals of unsigned percent change of mean tuning parameter values on days 2-41 from their corresponding value on day 1. Three electrodes were withheld form this analysis due to the small value of the tuning parameter for one or more reach directions on day 1. See text for details.

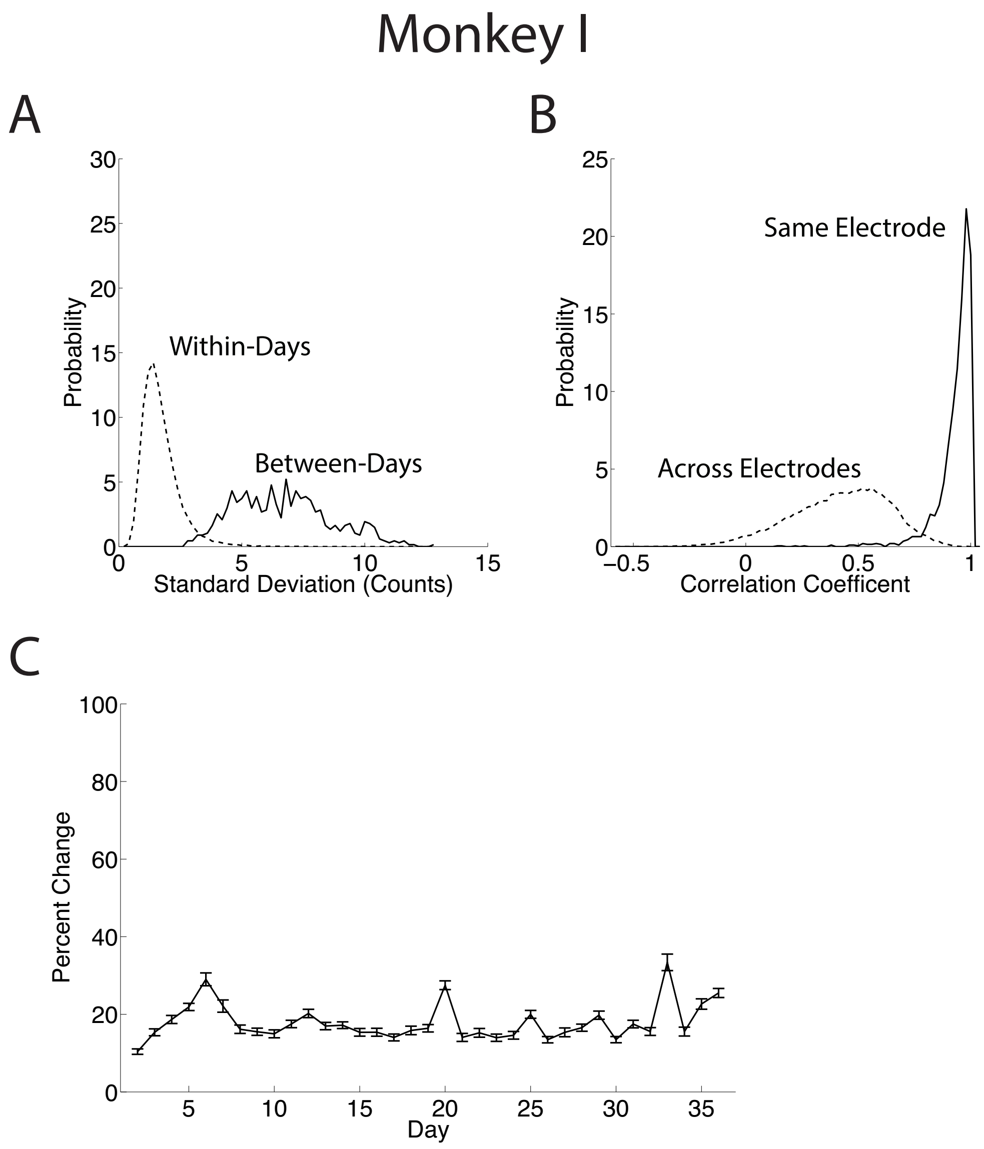


Figure 2.6: Summary of the statistical structure of changes in tuning parameters across days for monkey I. Panels A, B and C correspond to panels B, D and E in Fig. 2.5. No electrodes were withheld when calculating the average shown in panel C. For space reasons, panels showing representative parameters traces and the spread of values for a pair of tuning parameters are not shown for monkey I.

2.3.2 Evaluation of Classifier Stability with Prerecorded Data

We compare the performance of the SR, SR_S, standard retrained and standard non-retrained classifiers on the two prerecorded datasets described in section 2.2. The large number of days in each makes these ideal datasets for evaluating classifier stability.

The standard retrained classifier is trained fresh each day using the first 400 trials of the day. After training, the classifier is used to predict reach direction on each of the remaining trials of the day. Importantly, while true reach direction is known for each trial of the datasets and is used for training, this information is not made available to the classifier when predicting reach direction (testing) and is only used after a prediction is made to evaluate classifier accuracy.

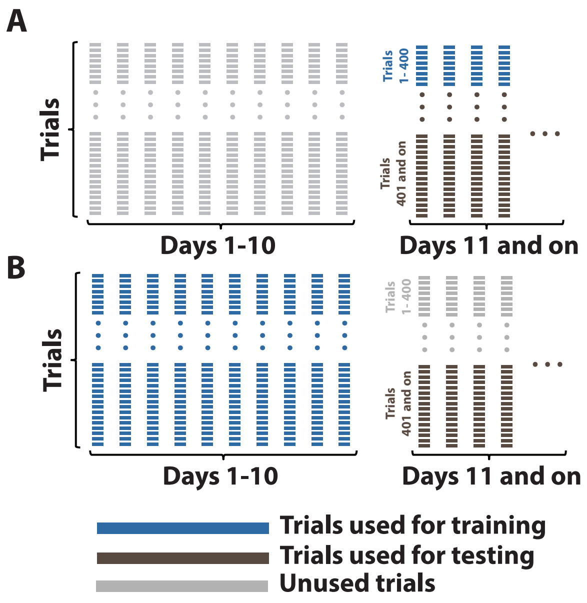


Figure 2.7: Partitioning of data when testing classifiers with (A) and without (B) daily retraining as described in Methods. Each column of rectangles represents an experimental day of collected data, and each rectangle represents a trial.

The SR, SR_S and standard non-retrained classifiers must be initially trained on a small number of days where neural activity is recorded and the desired reach direction for trials is known. After this they require no further training when classifying reach direction on days when only neural activity is available. For this analysis, all trials on the first 10 days of each dataset are used for training these classifiers. After this training, the classifiers are used to classify reach direction for trials on the remaining days of each dataset. To allow a fair head-to-head comparison with the standard retrained classifier, classification on each test day for these classifiers begins at trial 401 of each day. In this way, the set of trials that each classifier predicts reach direction for is the same on any given day. The breakup of data for training and testing each of the classifiers is shown in Fig. 2.7B.

When training all classifiers, any electrode with an average of less than two spike counts per trial on the training data (the first 400 trials of each day for the standard retrained classifier and all trials on days 1-10 for the standard non-retrained, SR and SR_S classifiers) is not used when subsequently decoding reach direction on test data. This step prevents these electrodes from causing problems if their average spike counts were to rise at a later point in time when performing classification. Critically, this can be done in clinical practice as the criterion for exclusion is based only on training data.

2.3.3 Performance of the Standard Classifier without Daily Retraining

We now examine the implications of tuning parameter drift on the standard non-retrained classifier. The performance of this classifier indicates what might happen if we simply used the standard classifier without retraining and did nothing to account for drifting tuning parameters. Of course, this is an offline analysis and subjects may be able to adapt their neural signals through cognitive strategies in a closed loop setting to improve BCI performance [76], but the results presented here may still be considered as a type of lower bound on performance without retraining.

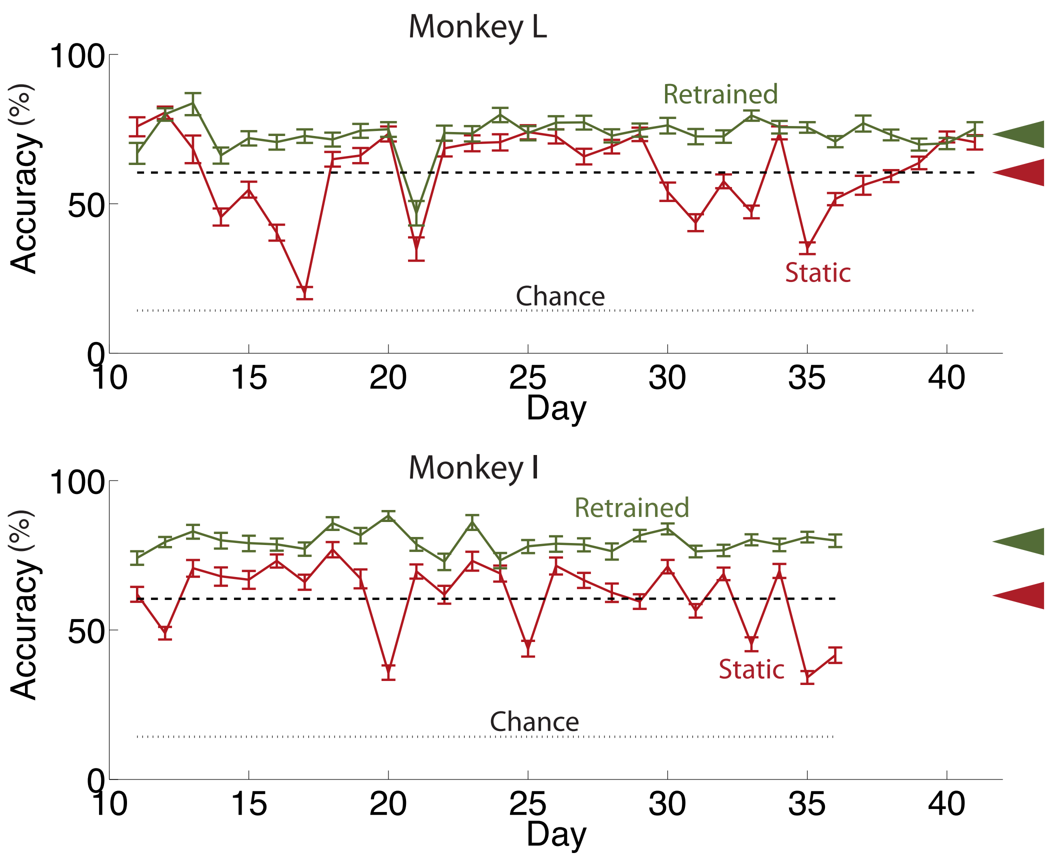


Figure 2.8: Daily accuracy values for the standard non-retrained and retrained classifiers for both subjects with 95% confidence intervals. The retrained classifier was retrained each day while the non-retrained classifier was trained on days preceding those decoded here and then held fixed. See Fig. 2.7 for a depiction of the partitioning of data when training and testing each classifier. The dashed line in each panel shows a linear fit of daily accuracy of the non-retrained classifier with test day. The slope of this fit is not significantly different from zero for either subject. Arrows on the right show overall accuracies of each classifier.

Examination of Fig. 2.8, which displays the average daily classification accuracy of the standard non-retrained classifier for both subjects, reveals two important findings. First, performance of the standard non-retrained classifier is substantially lower than that of the daily retrained classifier on most days. This motivates the development of classifiers which account for drifting tuning parameters. However, classification accuracy of the standard non-retrained classifier does not significantly decline with time, which constitutes the second finding. To quantify this claim, the slope of a linear fit to daily classification accuracy is not significantly different than zero for both subjects. Specifically, the 95% confidence interval for the slope, measured in units of percent accuracy per day, is $(-.006, .006)$ for monkey L and $(-.012, .001)$ for monkey I, which contains zero for both monkeys. While this finding might at first seem suprising, it is consistent with the finding above that tuning parameters do not show a systematic tendency to progressively wander from their values on day 1 but seem to vary within a relatively limited range.

Performance of the Self-Recalibrating Classifiers

Figs. 2.9 and 2.10 show the daily classification accuracies of the SR and SR_S classifiers for each subject. We emphasize that after these classifiers were trained on days 1-10 for a dataset, they were not provided with any information about the true reach direction of trials with which to retrain themselves when predicting reach direction on the following days. Instead, using the techniques described in section 2.2, the SR and SR_S classifiers simultaneously predicted reach direction and the daily value of tuning parameters from neural data alone. Therefore, the classification accuracies reported here are indicative of what might be achieved in the clinic without any further daily supervised retraining or recalibration after the system is initially trained.

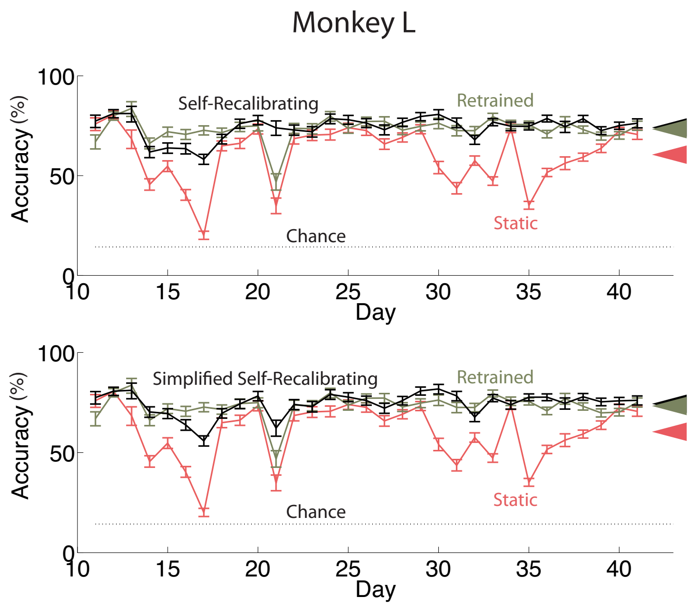


Figure 2.9: Daily performance of the self-recalibrating (top panel) and simplified self-recalibrating (bottom panel) classifiers with 95% confidence intervals for data from monkey L. Arrows on the right show overall accuracies of each classifier. Results for the standard classifier with and without daily retraining are reproduced from Fig. 2.8 for reference.

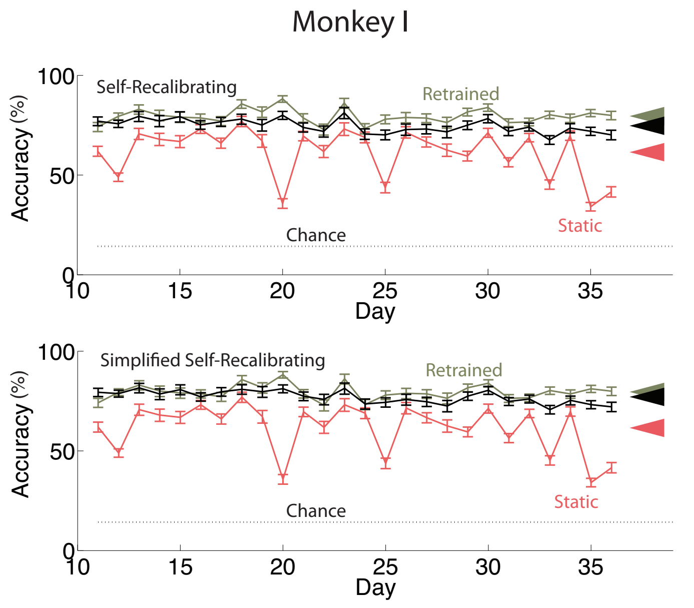


Figure 2.10: Daily performance of the self-recalibrating (top panel) and simplified self-recalibrating (bottom panel) classifiers with 95% confidence intervals for data from monkey I. Arrows on the right show overall accuracies of each classifier. Results for the standard classifier with and without daily retraining are reproduced from Fig. 2.8 for reference.

Table 2.1 compares the overall accuracies, calculated as the mean of the average daily accuracies, for all four classifiers considered in this chapter. The overall accuracies of the SR and SR_S classifiers were both 14% higher than that of the standard non-retrained classifier for monkey L and 13% and 16%, respectively, higher for monkey I. These performance improvements were statistically significant for both self-recalibrating classifiers and both subjects (One Sided Wilcoxon Signed-Rank Test, $p < 10^{-6}$ for both classifiers for monkey L, $p < 10^{-5}$ for both classifiers for monkey I). Further, both self-recalibrating classifiers nearly recovered the accuracy achieved by the standard retrained classifier. The overall accuracy of the SR and the SR_S classifiers were both 1% higher than that of the retrained classifier for monkey L and only 5% and 3% lower for monkey I. These differences were not statistically significant for either self-recalibrating classifier and monkey L (One Sided Wilcoxon Signed-Rank Test, $p > 0.1$ for SR classifier and $p > .05$ for the SR_S classifier), while the small decrease in accuracies observed on the data for monkey I were (One Sided Wilcoxon Signed-Rank Test, $p < 10^{-4}$ for SR classifier and $p < 10^{-2}$ for the SR_S classifier). To be clear, the self-recalibrating classifiers decoded the exact same trials, trials 401 and on for each test day, as the retrained classifier and achieved this performance without the benefit that the retrained classifier had of being retrained on the fully labeled data of the first 400 trials of each day.

An important consideration when evaluating the performance of a self-recalibrating classifier is the number of trials required to learn tuning parameter values. A self-recalibrating classifier is of little use if its performance starts near chance each day and then takes hundreds of trials to reach acceptable prediction accuracy. Fig. 2.11 shows the average classification accuracy achieved by each of the self-recalibrating classifiers throughout the course of a decoding day for both subjects. These accuracies were calculated by dividing all of the trials decoded during a test day into sequential bins of twenty trials each and then calculating the average prediction accuracy in each of these bins over all days of data for a subject. The number of trials in each day varied, and the number of bins that accuracies could be calculated over was limited by the day with the fewest number of trials for each subject. Monkey L had at least 386 trials for classification on each test day and monkey I had at least 752. Examination of the plots in Fig. 2.11 reveal that both classifiers start a decoding day well above chance levels of performance. Further, the SR classifier shows little evidence of a "burn-in" period, where accuracy might be lower as tuning parameter values are being learned. It is likely that if there were more days of data present and smaller bin sizes could be used in the analysis, a burn-in period of less than 20 trials might become apparent, but in any case the SR classifier rapidly achieves steady state performance. The SR_S classifier shows evidence of slightly decreased performance in the first bin of 20 trials for both subjects, and performance appears to be at asymptotic levels by trials 21-40.

Classifier	Monkey L	Monkey I
Non-Retrained	$60 \pm 5\%$	$62 \pm 5\%$
Retrained	$73 \pm 2\%$	$80 \pm 2\%$
Self-Recalibrating	$74 \pm 2\%$	$75 \pm 1\%$
Simplified Self-Recalibrating	$74 \pm 2\%$	$77 \pm 1\%$

Table 2.1: Average daily accuracy with 95% confidence intervals for the standard non-retrained, retrained and both self-recalibrating classifiers for both subjects.

Fig. 2.12 displays tuning parameter estimates on two electrodes produced by both self-recalibrating classifiers on the first five days of test data for monkey L. Results for monkey I were similar. For this analysis, estimates for true tuning parameters were obtained by convolved spikes with a Gaussian kernel with a standard deviation of 100 trials as described in *Methods* with one exception: days were processed separately and not concatenated together. The SR classifier produces a distribution over tuning parameter values on each trial (shown as the distributions over $\mu_{d,e,j=1}$ and $\mu_{d,e,j=2}$ in Fig. 2.2), and the mean of these distributions, $\mathbb{E}[\mu_{d,e,j}|\{x_{d,s}\}_{s=1}^{t}]$, was used to plot the traces in Fig. 2.12. Even though there are seven tuning parameters per electrode, traces for only two tuning parameters per electrode are shown for visual clarity. Characteristics of the way both classifiers predict tuning parameters become apparent with inspection of the plots in Fig. 2.12. First, on all days estimates for tuning parameters on the same electrode move up and down together. This immediately follows from the modeling assumption that tuning parameters on a given electrode are yoked to a single base value, $b_{d,e}$, in eq. 2.5. Second, tuning parameter estimates vary more strongly at the beginning than at the end of a decoding session. This can be observed in the steep change in tuning parameter values at the beginning of certain days, such as day 14 for electrode 17 in Fig. 2.12. Intuitively, the reason for the reduced variability in tuning parameter estimates for both decoders is that confidence in estimates of base values increases as more neural activity is observed. Mathematically, this corresponds to less uncertainty in the distribution of base values, $P(b_d|\{x_{d,s}\}_{s=1}^t)$, for the SR classifier and an increasing value of $n_{d,t}$ for the SR_S classifier.

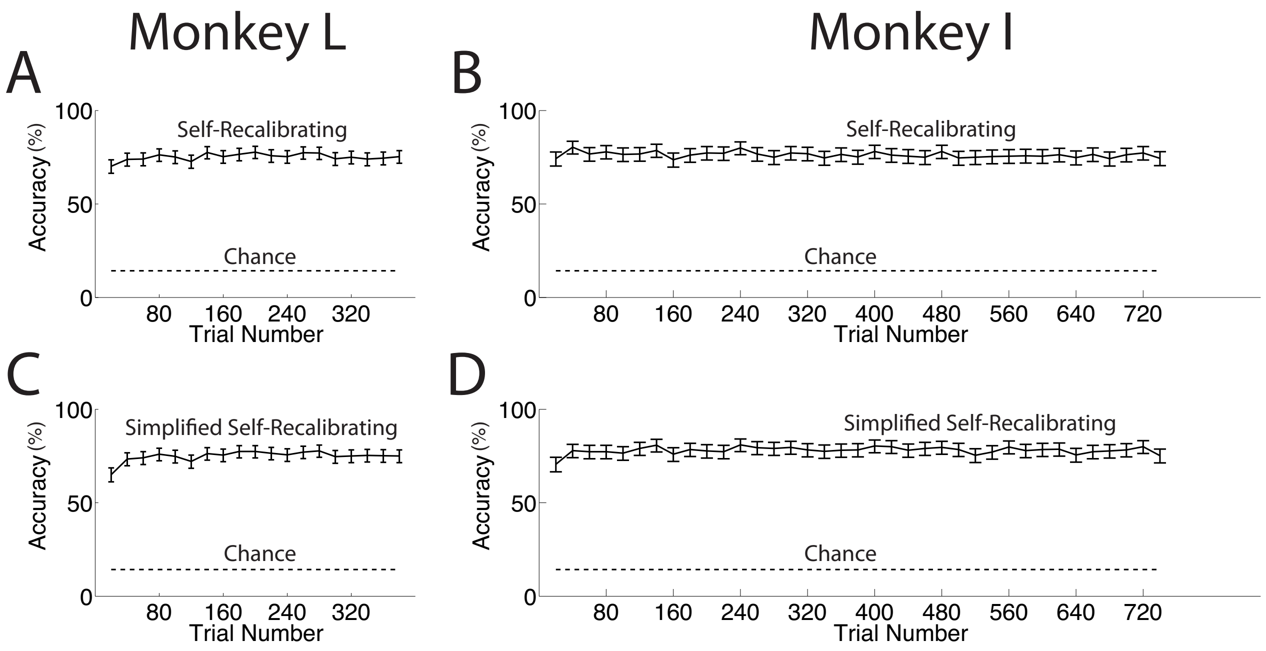


Figure 2.11: Average accuracy throughout a decoding session for the SR and SR_S classifiers for both subjects with 95% confidence intervals. Accuracy values are calculated as the average classification accuracy in sequential bins of 20 trials over all test days for a subject. There were 386 or more trials for classification on each test day for monkey L and 752 or more on each test day for monkey I. Panels A and C show results for monkey L, while panels B and D show results for monkey I.

The "spikes" in the parameter estimates for an electrode produced by SR classifier visible in both figures occur when that electrode is flagged as behaving erratically by the mechanism for detecting anomalous spike counts, which may occur after the threshold for an electrode is reset, as described in section 2.2.4. When this occurs, the uncertainty around the base value for that electrode is increased. This allows spike counts collected immediately after an electrode is flagged to strongly influence parameter estimates, and just as at the beginning of a day, multiple trials have to pass before the uncertainty for the base value of a flagged electrode is reduced and parameter estimates vary less. The reader may notice that the plotted electrodes are flagged as behaving erratically multiple times within the same day. As explained in section 2.2.4, the false positive rate of the flagging mechanism was set to 1%; therefore, the frequency of parameter resets seen in Fig. 2.12 should not be understood to indicate the true frequency of anomalous events. While we do not show results for this, in practice the SR classifier appears to be reasonably robust without this mechanism to detect and compensate for electrodes with anomalous spike counts. In fact, classification accuracy of the SR classifier was markedly improved on only three days for monkey L and on no days for monkey I by its inclusion.

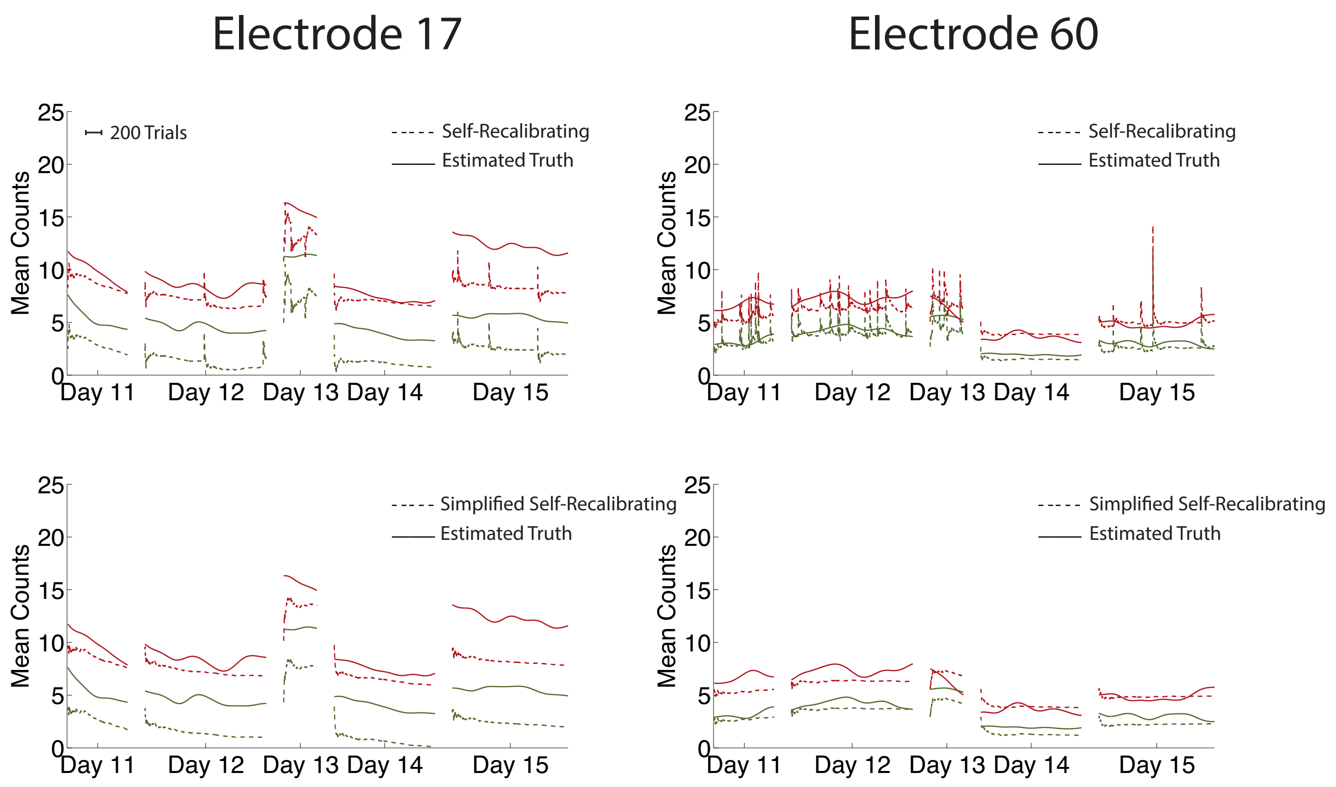


Figure 2.12: Estimates of tuning parameters on two representative electrodes (left and right columns) produced by the SR (top) and SR_S (bottom) classifiers on the first five days of test data for monkey L compared to smoothed estimates obtained by convolving spikes with a Gaussian kernel. The "spikes" in the tuning parameter estimate of the SR classifier occur when the mechanism for detecting anomalous spike counts is triggered and the uncertainty around the parameter estimates for an electrode is reset. The false positive rate for detecting anomalous spikes was set to approximately 1%, and the rate of spikes seen in this plot should not be understood as the true rate of anomalous events.

2.4 Discussion

The development of self-recalibrating classifiers for intracortical BCI has received little previous attention. In this chapter, we first characterized properties of tuning parameter drift which can be leveraged for the design of self-recalibrating classifiers. We provided evidence that between-day tuning parameter drift is substantially larger than within-day drift and that tuning parameters on the same electrode drift in a highly correlated manner. Further, tuning parameter drift appears constrained across time. This is consistent with the performance of a non-retrained classifier, which is unstable from day-to-day but does not tend to decline with time. We developed two novel, self-recalibrating classifiers which leverage the insights gained into tuning parameter drift to maintain stable performance over periods of 31 and 26 days of prerecorded data. This results in an approximate 15% improvement in overall accuracy compared to the standard classifier without daily retraining. This improvement was large enough that both self-recalibrating classifiers nearly recovered the accuracy achieved by the standard classifier when it was retrained daily.

The self-recalibrating classifiers presented in this chapter train themselves daily without knowledge of true reach directions. This is the first report of such classifiers for intracortical BCI to the authors' knowledge. While these results must be verified in a closed-loop setting, they may represent a significant step towards the clinical viability of intracortical BCI systems. Indeed, future systems patterned off of the ideas in this chapter would be ready for use almost immediately after being powered on each day as threshold levels can be set via an automated method and the classifiers update their parameters autonomously in the background during normal BCI use.

The classifiers presented here require foreknowledge of the window in time during which a user is making a decision. Previous work has demonstrated the utility of BCI systems subject to this constraint [77]. Further, self-paced BCI systems can be developed by combining the self-recalibrating classifiers presented here with previously published algorithms to determine at each point in time whether the user is intending to make a decision or not [50,51].

The two self-recalibrating classifiers performed very similarly. Is one to be preferred over the other for clinical application? The SR_S classifier adds little computational cost to the standard BCI classifier. This has important implications for resource constrained systems, such as implantable medical devices. Computational hardware which supports the standard classifier will also likely support an upgrade to the simplified self-recalibrating classifier. However, the SR classifier may still be preferred in two scenarios. First, as shown in Fig. 2.11, the SR classifier requires fewer trials to reach stable levels of performance at the beginning of a decoding day. Second, the SR_S classifier implicitly assumes that a user makes reaches in all directions with equal frequency. If this assumption is broken, which may occur if a user reaches predominantly in one direction, tuning parameter estimation will suffer. The SR classifier bypasses this shortcoming by calculating the probability of reaching in each direction while estimating tuning parameters for each trial. In addition to these considerations, the SR classifier makes modeling assumptions explicit. Changes to these assumptions can be carried out in a straight forward manner with the SR classifier, and it not obvious how this might be done with the SR_S classifier.

Tuning parameter estimates produced by both self-recalibrating classifiers could differ significantly from those estimated as ground truth. However, this did not prevent either classifier from nearly reproducing the decoding accuracy of the daily retrained standard classifier. While this may seem surprising, it is likely classification accuracy is robust to small deviations in estimated tuning parameters as long as their relative ordering is maintained. Our models ensure this by referencing all tuning parameters on an electrode to a single base value. Preliminary analyses with more complicated models which did not force tuning parameters on the same electrode to move in perfect lockstep and also modeled drift in tuning parameters within days did not reveal markedly improved decoding performance (data not shown). Therefore, as the ultimate goal of the work in this chapter is stable classification and not tuning parameter estimation, we believe the simpler models are to be preferred.

Our decoders also do not carry information about tuning parameters from one decoding day to the next. However, this does not significantly hinder decoding performance. This may seem surprising at first as Fig. 2.5A indicates that on some pairs of days the previous day's tuning parameter values may be predictive of the next day's values. However, we note that this is not the case for all days. In Fig. 2.5A, for example, there is a large jump in the value of parameters between days 4 and 5. Thus, on same days using information about the decoding parameters from the previous day would likely help the decoder learn the correct tuning parameters for the present day more quickly, and on some others it could hinder learning. Additionally, as shown in Figs. 11B and 11D, both decoders learn tuning parameter values quickly enough to allow average decoding performance to reach asymptotic levels in tens of trials. Thus, the improvement in decoding accuracy that could be realized by learning tuning parameters more quickly is fairly minimal. Finally, formally modeling a time series of base values between days would make our EM algorithm for the SR classifier more complicated as the distribution of base values conditioned on observed data would no longer factor as a product of the distribution of base values on individual days (see equation eq. A.1 in appendix A), and additional parameters for a model of transitions in base values between days would need to be learned. Though these reasons motivate the simplification to not carry tuning parameter information between days, we acknowledge this may be interesting future work.

An important aim of this chapter was to identify properties of the drift in intracortical BCI signals that could be exploited for the development of simplified classifiers. The finding that tuning parameters on the same electrode drift in a highly correlated manner reduces the dimensionality of the space that tuning parameters must be tracked in as algorithms can track a single base value instead all individual parameters on an electrode. High correlations in the drift of tuning parameters on the same electrode may be consistent with changes in baseline firing rates. Changes in baseline could be driven by changes in recorded action potential amplitude [64], micromovements of the recording array relative to cortical tissue [66] and daily fluctuations in the value of the threshold that was used for spike detection. Whatever the mechanisms which underlie this behavior, robust classification was achieved by the relatively simple SR_S classifier. The success of this simple approach is tantalizing because it suggests that any BCI algorithm which can be parameterized by variables which are referenced to the average spike counts of electrodes could be made self-recalibrating by following a recipe similar to that presented in section 2.2.5 for the SR_S classifier.

Another important finding was that tuning parameters drift much more between than within days. This suggests that self-recalibrating classifiers can be simplified to capture only between-day changes, as was done in this chapter. Recently Perge et al. investigated intracortical BCI signal stability within the context of decoding continuous control signals (i.e., regression) in closed loop human experiments [78]. While they do not explicitly compare within-day changes to between-day changes in BCI signals, they do report reduced decoding accuracy due to within-day changes in neural spike rates. The differing conclusions may be explained by the relative sensitivity of regressors and classifiers to changes in BCI signals. Even small signal changes in BCI signals will likely be reflected in the decoded output of a regressor, whereas the performance of a classifier will only be affected if the relative ordering of the probabilities assigned to each class (reach direction in this work) are altered. This should lend caution when applying our findings on tuning parameter drift to the continuous decoding scenario. While tuning parameters do move up and down together between days on the same electrode, other aspects of tuning, such as modulation depth, as shown in appendix A, can also change. For the purposes of developing self-recalibrating classifiers for our datasets, we were able to simplify our classifiers and ignore these other aspects of drift. However, it may be that these simplifications would be more detrimental in the context of regression. This is not to degrade the importance of the present study. As discussed in the introduction, BCI classifiers offer unique and important clinical promise, and insights that allow for the unique simplification of BCI classifiers may facilitate their use in non-laboratory settings.

Recently Flint et al. have examined BCI performance of a non-retrained continuous decoder of arm movements in a closed loop setting [79] when either local field potentials or threshold crossings were used for control signals. We focus on their results for threshold crossings. Two findings of their work are of direct relevance to the present study. First, in an offline analysis of hand-control data, they report highly variable day-to-day performance but little change in average performance with time as decoders age after the initial day they were trained on. This is consistent with our similar finding for a discrete decoder. Second, when they measure correlation coefficients between a variety of measures of closed loop decoder performance and time, they show non-degrading performance over time for one subject and a mild degradation of performance for a second. Beyond these overall trends, day-to-day variability in performance is still apparent. While this may be partly due to variability in subject motivation, it is also quite possible that the use of self-recalibrating classifiers could have provided reduced day-to-day variability in performance in this closed loop setting.

There are at least two possible reasons tuning parameter drift is constrained across time, as we observed in this chapter. First, it may be the case that tuning parameters are fixed across days but our estimates varied from day to day due to fluctuations in threshold settings based on a finite sample (two seconds) of data. However, this likely does not entirely explain the changes in tuning parameters that we observe because the relationship between multiunit activity and arm kinematics can change even when threshold settings are held fixed across days [79]. A second tantalizing possibility is that micromovements of the recording array might underlie such tuning parameter changes so that electrodes are in a random position within a relatively stable neural environment. If this is the case, the development of recording arrays that maintain a more fixed position relative to brain tissue may hold promise for increasing the fundamental stability of signals for intracortical BCI use.

Chapter 3

Deterministic Symmetric Positive Semidefinite Matrix Completion

3.1 Background

We now move from an applied setting to theory and consider the problem of recovering a low-rank, symmetric positive semidefinite (SPSD) matrix when only some of it's, possibly noise corrupted, entries are observed. Covariance matrices are examples of SPSD matrices which often appear in practice. If the activity of a population of neurons is truly low-dimensional, the covariance matrix for the population of neurons will be low-rank. In the rest of this thesis we will do more than simply recover covariance matrices, but for the moment, this chapter can be motivated by considering the problem of estimating the full covariance matrix for a population of neurons when we can only make pair-wise recordings from some but not all pairs of neurons in the population.

There are multiple scenarios, in addition to the one just mentioned, where we might wish to reconstruct a symmetric positive semidefinite (SPSD) matrix from a sampling of its entries. In multidimensional scaling, for example, pairwise distance measurements are used to form a kernel matrix and PCA is performed on this matrix to embed the data in a low-dimensional subspace. However, it may not be possible to measure pairwise distances for all variables, rendering the kernel matrix incomplete. More generally, SPSD matrices in the form of Gram matrices play a key role in a broad range of machine learning problems such as support vector machines [80], Gaussian processes [81] and nonlinear dimensionality reduction techniques [82] and the reconstruction of such matrices from a subset of their entries is of general interest.

In real world scenarios, the constraints that make it difficult to observe a whole matrix often also constrain which particular entries of a matrix are observable. In such settings, existing matrix completion results, which assume matrix entries are revealed in an unstructured, random manner [24–26, 29, 30, 83–85] or the ability to finely query individual entries of a matrix in an adaptive manner [86, 87] might not be applicable. This motivates us to examine the problem of recovering a SPSD matrix from a given, deterministic set of its entries. In particular we focus on reconstructing a SPSD matrix from a revealed set of its principal submatrices.

Recall that a principal submatrix of a matrix is a submatrix obtained by symmetrically intersecting rows and columns of the original matrix. When individual entries of a matrix are formed by pairwise measurements between experimental variables, principal submatrices are a natural way to formally capture how entries are revealed. See Fig. 3.1 for two example principal submatrices.

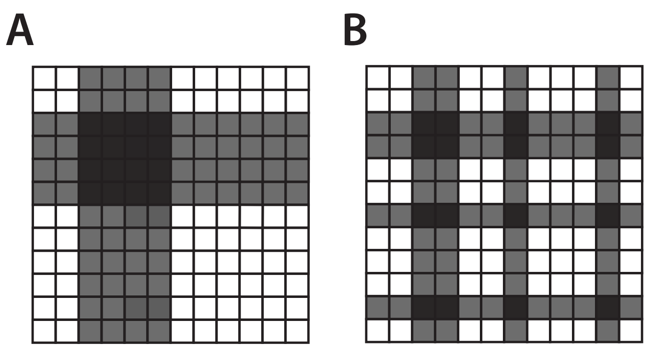


Figure 3.1: Two examples of principal submatrices, which are formed by symmetrically intersecting rows and columns. The set of intersected rows and columns is shown in the lighter grey, with the darker regions representing the principal submatrices. (A) A principal submatrix formed by intersecting adjacent rows and columns. (B) A principal submatrix formed from intersecting non-adjacent rows and columns.

Sampling principal submatrices also allows for an intuitive method of matrix reconstruction. As shown in Fig. 3.1, any $n \times n$ rank r SPSD matrix A can be decomposed as $A = CC^T$ for some $C \in \mathbb{R}^{n \times r}$. Any principal submatrix of A can also be decomposed in the same way. Further, if ρ_i is an ordered set indexing the the i^{th} principal submatrix of A, it must be that $A(\rho_i, \rho_i) = C(\rho_i, :)C(\rho_i, :)^T$.[1] This suggests we can decompose each $A(\rho_i, \rho_i)$ to learn the the rows of C and then reconstruct A from the learned C, but there is one complication. Any matrix, $C(\rho_i, :)$, such that $A(\rho_i, \rho_i) = C(\rho_i, :)C(\rho_i, :)^T$, is only defined up to an orthonormal transformation. The naïve algorithm just suggested has no way of ensuring the rows of C learned from two different principal submatrices are consistent with respect to this degeneracy. Fortunately, the situation is easily remedied if the principal submatrices in question have some overlap, so that the $C(\rho_i, :)$ matrices have some rows that map to each other. Under appropriate conditions explored below, we can learn unique orthonormal transformations rendering these rows equal, allowing us to align the $C(\rho_i, :)$ matrices to learn a proper C.

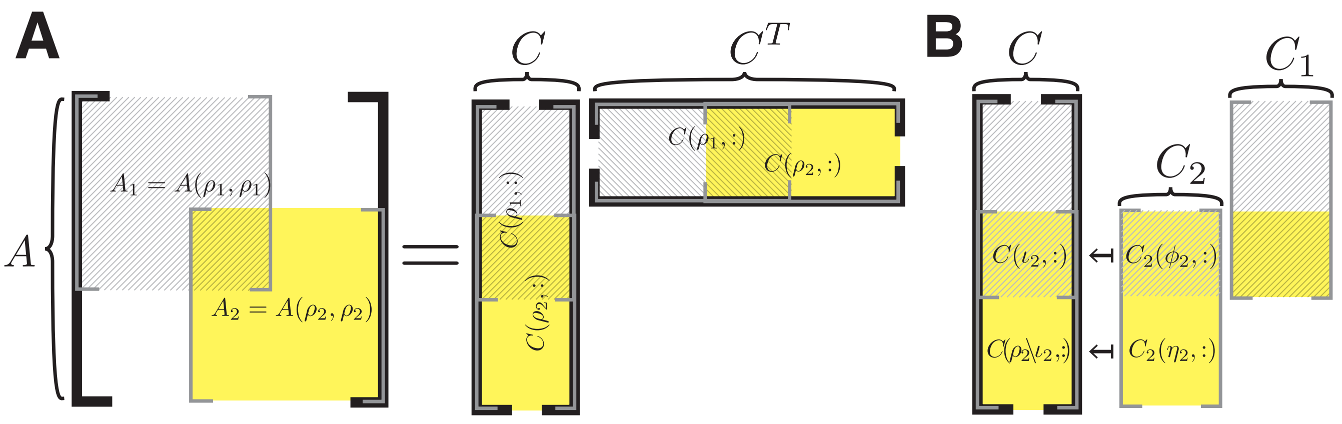


Figure 3.2: (A) An example A matrix with two principal submatrices, showing the correspondence between $A(\rho_l, \rho_l)$ and $C(\rho_l, :)$. (B) Mapping of C_1 and C_2 to C, illustrating the role of ι_l, ϕ_l and η_l.

Contributions In this chapter, we make the following contributions.

1. We prove sufficient conditions, which are also necessary in certain situations, for the exact recovery of a SPSD matrix from a given set of its principal submatrices.

[1] Throughout this chapter we will use MATLAB indexing notation, so $C(\rho_i, :)$ is the submatrix of C made up of the rows indexed by the ordered set ρ_i.

2. We present a novel algorithm which exactly recovers a SPSD matrix when the sufficient conditions are met.
3. The algorithm is generalized when the set of observed principal submatrices of a matrix are corrupted by noise. We present a theorem guaranteeing a bound on reconstruction error.

3.1.1 Related Work

The low rank matrix completion problem has received considerable attention since the work of Candès and Recht [28] who demonstrated that a simple convex problem could exactly recover many low-rank matrices with high probability. This work, as did much of what followed (e.g., [24–26]), made three key assumptions. First, entries of a matrix were assumed to be uncorrupted by noise and, second, revealed in a random, unstructured manner. Finally, requirements, such as incoherence, were also imposed to rule out matrices with most of their mass concentrated in a only a few entries.

These assumptions have been reexamined and relaxed in additional work. The case of noisy observed entries has been considered [29,30,83–85]. Others have reduced or removed the requirements for incoherence by using iterative, adaptive sampling schemes [86,87]. Finally, recent work [88,89] has considered the case of matrix recovery when entries are selected in a deterministic manner.

The work in this chapter considerably differs from this earlier work. Our applications of interest allow us to assume much structure, i.e., that matrices are SPSD, which our algorithm exploits, and our sufficient conditions make no appeal to incoherence. Our work also differs from previous results for deterministic sampling schemes (e.g., [88,89]), which do not consider noise nor provide sufficient conditions for exact recovery, instead approaching the problem as one of matrix approximation.

Previous work has also considered the problem of completing SPSD matrices of any [90] or low rank [91,92]. Our work to identify conditions for a unique completion of a given rank can be viewed as a continuation of this work where our sufficient and necessary conditions can be understood in a particularly intuitive manner due to our sampling scheme. Finally, the Nyström method [93] is a well known technique for approximating a SPSD matrix as low rank. It can also be applied to the matrix recovery problem, and in the noiseless case, sufficient conditions for exact recovery are known [94]. However, the Nyström method requires sampling full columns and rows of the original matrix, a sampling scheme which may not be possible in many of our applications of interest.

3.2 Preliminaries

3.2.1 Deterministic Sampling for SPSD Matrices

We denote the set of index pairs for the revealed entries of a matrix by Ω. Formally, an index pair, (i, j), is in Ω if and only if we observe the corresponding entry of an $n \times n$ matrix so that $\Omega \subset [n] \times [n]$.[2] In this work, we assume Ω indexes a set of principal submatrices of a matrix. Let $\Omega_l \subseteq \Omega$ indicate a subset of Ω. If Ω_l indexes a principal submatrix of a matrix, it can be compactly described by the unique set of row (or equivalently column) indices it contains. Let $\boldsymbol{\rho}\{\Omega_l\} = \{i | (i,j) \in \Omega_l\}$ be the set of row indices contained in Ω_l. For compactness, let $\rho_l = \boldsymbol{\rho}\{\Omega_l\}$. Finally, let $|\cdot|$ indicate cardinality. Then, for an $n \times n$ matrix, A, of rank $r > 0$ we make the following assumptions on Ω.

(A1) $\boldsymbol{\rho}\{\Omega\} = [n]$.

(A2) There exists a collection $\Omega_1, \ldots, \Omega_k$ of subsets of Ω such that $\Omega = \cup_{l=1}^{k} \Omega_l$, and for each Ω_l, $(i,i) \in \Omega_l$ and $(j,j) \in \Omega_l$ if and only if $(i,j) \in \Omega_l$ and $(j,i) \in \Omega_l$.

(A3) There exists a collection $\Omega_1, \ldots, \Omega_k$ of subsets of Ω such that $A2$ holds and if $k > 1$, there exists an ordering $\tau_1, \ldots, \tau_k$ such that for all $i \geq 2$, $|\rho_{\tau_i} \cap \left(\cup_{j=1}^{i-1} \rho_{\tau_j} \right)| \geq r$.

The first assumption ensures Ω indexes at least one entry for each row of A. Assumption $A2$ requires that Ω indexes a collection of principal submatrices of A, and $A3$ allows for the possible alignment of rows of C (recall, $A = CC^T$) estimated from each principal submatrix.

[2] We use the notation, $[n]$ to indicate the set $\{1, \ldots, n\}$.

Algorithm 1 SPSD Matrix Recovery $(r, \{\tilde{E}_l, \tilde{\Lambda}_l, \tau_l, \rho_l, \phi_l, \iota_l, \eta_l\}_{l=1}^{k})$

```
Initialize $\hat{C}$ as a $n \times r$ matrix.
  1. $\hat{C}(\rho_{\tau_1}, :) \leftarrow \tilde{E}_{\tau_1}(:, 1:r)\tilde{\Lambda}_{\tau_1}^{1/2}(1:r, 1:r)$
  2. For $l \in \{2, \dots, k\}$
      (a) $\hat{C}_l \leftarrow \tilde{E}_{\tau_l}(:, 1:r)\tilde{\Lambda}_{\tau_l}^{1/2}(1:r, 1:r)$
      (b) $\hat{W}_l \leftarrow \text{argmin}_{WW^T=I} \|\hat{C}(\iota_l, :) - \hat{C}_l(\phi_l, :)W\|_F^2$
      (c) $\hat{C}(\rho_{\tau_l} \setminus \iota_l, :) \leftarrow \hat{C}_l(\eta_l, :)\hat{W}_l$
  3. Return $\hat{A} = \hat{C}\hat{C}^T$
```

3.2.2 Additional Notation

Denote the set of real, $n \times n$ SPSD matrices by $\mathcal{S}_+^n$, and let $A \in \mathcal{S}_+^n$ be the rank $r > 0$ matrix to be recovered. For the noisy case, $\tilde{A}$ will indicate a perturbed version of A. We will use the notation A_l to indicate the principal submatrix of a matrix A indexed by Ω_l.

Denote the eigendecomposition of A as $A = E\Lambda E^T$ for the diagonal matrix $\Lambda \in \mathbb{R}^{r \times r}$ containing the non-zero eigenvalues of A, $\lambda_1 \geq \ldots \geq \lambda_r$, along its diagonal and the matrix $E^{n \times r}$ containing the corresponding eigenvectors of A in its columns. Let n_l denote the size of A_l and r_l the rank. Because A_l is a principal submatrix of A, it follows that $A_l \in \mathcal{S}_+^{n_l}$. Denote the eigendecomposition of each A_l as $A_l = E_l \Lambda_l E_l^T$ for the matrices $\Lambda_l \in \mathbb{R}^{r_l \times r_l}$ and $E_l \in \mathbb{R}^{n_l \times r_l}$. We add tildes to the appropriate symbols for the eigendecomposition of $\tilde{A}$ and its principal submatrices.

Finally, let $\iota_l = \rho_{\tau_l} \cap (\cup_{j=1,\ldots,l-1} \rho_{\tau_j})$ be the intersection of the indices for the l^{th} principal submatrix with the indices of the all of the principal submatrices ordered before it. Let C_l be a matrix such that $C_l C_l^T = A_l$. If A_l is a principal submatrix of A there will exist some C_l such that $C(\rho_l, :) = C_l$. For such a C_l, let ϕ_l be an index set that assigns the rows of the matrix $C(\iota_l, :)$ to their location in C_l, so that $C(\iota_l, :) = C_l(\phi_l, :)$ and let η_l assign the rows of $C(\rho_l \setminus \iota_l, :)$ to their location in C_l, so that $C(\rho_l \setminus \iota_l, :) = C_l(\eta_l, :)$. The role of ρ_l, ι_l, η_l and ϕ_l is illustrated for the case of two principal submatrices with $\tau_1 = 1$, $\tau_2 = 2$ in Figure 3.1.

3.3 The Algorithm

Before establishing a set of sufficient conditions for exact matrix completion, we present our algorithm. Except for minor notational differences, the algorithms for the noiseless and noisy matrix recovery scenarios are identical, and for brevity we present the algorithm for the noisy scenario.

Let Ω sample the observed entires of $\tilde{A}$ so that $A1$ through $A3$ hold. Assume each perturbed principal submatrix, $\tilde{A}_l$, indexed by Ω is SPSD and of rank r or greater. These assumptions on each $\tilde{A}_l$ will be further explored in section 3.5. Decompose each $\tilde{A}_l$ as $\tilde{A}_l = \tilde{E}_l \tilde{\Lambda}_l \tilde{E}_l^T$, and form a rank r matrix $\hat{C}_l$ as $\hat{C}_l = \tilde{E}_l(:, 1:r)\tilde{\Lambda}_l^{1/2}(1:r, 1:r)$.

The rows of the $\hat{C}_l$ matrices contain estimates for the rows of C such that $A = CC^T$, though rows estimated from different principal submatrices may be expressed with respect to different orthonormal transformations. Without loss of generality, assume the principal submatrices are labeled so that $\tau_1 = 1, \ldots, \tau_k = k$. Our algorithm begins to construct $\hat{C}$ by estimating $\hat{C}(\rho_1, :) = \hat{C}_1$. In this step, we also implicitly choose to express $\hat{C}$ with respect to the basis for $\hat{C}_1$. We then iteratively add rows to $\hat{C}$, for each $\hat{C}_l$ adding the rows $\hat{C}_l(\eta_l, :)$ to $\hat{C}$. To estimate the orthornormal transformation to align the rows of $\hat{C}_l$ with the rows of $\hat{C}$ estimated in previous iterations, we solve the following optimization problem

$$\hat{W}_l = \underset{WW^T=I}{\text{argmin}} \left\| \hat{C}(\iota_l, :) - \hat{C}_l(\phi_l, :)W \right\|_F^2. \tag{3.1}$$

In words, equation 3.1 estimates $\hat{W}_l$ so that the rows of $\hat{C}_l$ which overlap with the previously estimated rows of $\hat{C}$ match as closely as possible. In the noiseless case, (3.1) is equivalent to $\hat{W}_l = W : \hat{C}(\iota_i, :) - \hat{C}_l(\phi_i, :)W = 0$.

Equation 3.1 is known as the Procrustes problem and is non-convex, but its solution can be found in closed form and sufficient conditions for its unique solution are known [95].

After learning $\hat{W}_l$ for each $\hat{C}_l$, we build up the estimate for $\hat{C}$ by setting $\hat{C}(\rho_l \setminus \iota_l, :) = \hat{C}_l(\eta_l, :)\hat{W}_l$. This step adds the rows of $\hat{C}_l$ that do not overlap with those already added to $\hat{C}$ to the growing estimate of $\hat{C}$. If we process principal submatrices in the order specified by $A3$, this algorithm will generate a complete estimate for $\hat{C}$. The full matrix $\hat{A}$ can then be estimated as $\hat{A} = \hat{C}\hat{C}$. The pseudocode for this algorithm is given in Algorithm 1.

3.4 The Noiseless Case

We begin this section by stating one additional assumption on A.

(A4) There exists a collection $\Omega_1, \ldots, \Omega_k$ of subsets of Ω such that $A2$ holds and if $k > 1$, there exists an ordering $\tau_1, \ldots, \tau_k$ such that the rank of $A(\iota_l, \iota_l)$ is equal to r for each $l \in \{2, \ldots, k\}$.

In Theorem 2 we show that $A1$ - $A4$ are sufficient to guarantee the exact recovery of A. Conditions $A1$ - $A4$ can also be necessary for the unique recovery of A by any method, as we show next in Theorem 1. Theorem 1 may at first glance appear quite simple, but it is a restatement of Lemma 20 in the appendix, from which more general necessity results can be derived. Specifically, Corollary 21 in the appendix can be used to establish necessity conditions for recovering A from a set of its principal submatrices which can be aligned in a overlapping sequence (e.g., submatrices running down the diagonal of A), which might be encountered when constructing a covariance matrix from sequentially sampled subgroups of variables. Corollary 22 establishes a similar result when there exists a set of principal submatrices which have no overlap among themselves but all overlap with one other submatrix not in the set, and Corollary 23 establishes that it is often sufficient to find just one principal submatrix that obeys certain conditions with respect to the rest of the sampled entries of the matrix to certify the impossibility of matrix completion. This last corollary in fact applies even when the rest of the sampled entries do not fall into a union of principal submatrices of the matrix.

Theorem 1. *Let $\Omega \neq [n] \times [n]$ index A so that $A2$ holds for some $\Omega_1 \subseteq \Omega$ and $\Omega_2 \subseteq \Omega$. Assume $\textbf{rank}\{A(\rho_1, \rho_1)\} = \textbf{rank}\{A(\rho_2 \setminus \iota_2, \rho_2 \setminus \iota_2)\} = r$. Then $A1$, $A3$ and $A4$ must hold with respect to Ω_1 and Ω_2 for A to be recoverable by any method.*

The proof can be found in the appendix. Here we briefly provide the intuition. Key to understanding the proof is recognizing that recovering A from the set of entries indexed by Ω is equivalent to learning a matrix C from the same set of entries such that $A = CC^T$. If $A1$ is not met, a complete row and the corresponding column of A is not sampled, and there is nothing to constrain the estimate for the corresponding row of C. If $A3$ and $A4$ are not met, we can construct a C such that all of the entries of the matrices A and CC^T indexed by Ω are identical yet $A \neq CC^T$.

We now show that our algorithm can recover A as soon as the above conditions are met, establishing their sufficiency.

Theorem 2. *Algorithm 1 will exactly recover A from a set of its principal submatrices indexed by $\Omega_1, \ldots, \Omega_k$ which meets conditions $A1$ through $A4$.*

The proof, which is provided in the appendix, shows that in the noiseless case, for each principal submatrix, A_l of A, step 2a of Algorithm 1 will learn an exact $\hat{C}_l$ such that $A_l = \hat{C}_l\hat{C}_l^T$. Further, when assumptions $A3$ and $A4$ are met, step 2b will correctly learn the orthonormal transformation to align each $\hat{C}_l$ to the previously added rows of $\hat{C}$. Therefore, progressive iterations of step 2 correctly learn more and more rows of a unified $\hat{C}$. As the algorithm progresses, all of the rows of $\hat{C}$ are learned and the entirety of A can be recovered in step 3 of the algorithm.

It is instructive to ask what we have gained or lost by constraining ourselves to sampling principal submatrices. In particular, we can ask how many individual entries must be observed before we can recover a matrix. A SPSD matrix has at least nr degrees of freedom, and we would not expect any matrix recovery method to succeed before at least this many entries of the original matrix are revealed. The next theorem establishes that our sampling scheme is not necessarily wasteful with respect to this bound.

Theorem 3. *For any rank $r > 0$ matrix $A \in \mathcal{S}_+^n$ there exists a Ω such that $A1 - A3$ hold and*

$$|\Omega| \leq n(2r+1).$$

Of course, this work is motivated by real-world scenarios where we are not at the liberty to finely select the principal submatrices we sample, and in practice we may often have to settle for a set of principal submatrices which sample more of the matrix. However, it is reassuring to know that our sampling scheme does not necessarily require a wasteful number of samples.

We note that assumptions $A1$ through $A4$ have an important benefit with respect to a requirement of incoherence. Incoherence is an assumption about the entire row and column space of a matrix and cannot be verified to hold with only the observed entries of a matrix. However, assumptions $A1$ through $A4$ can be verified to hold for a matrix of known rank using its observed entries. Thus, it is possible to verify that these assumptions hold for a given Ω and A and provide a certificate guaranteeing exact recovery before matrix completion is attempted.

3.5 The Noisy Case

We analyze the behavior of Algorithm 1 in the presence of noise. For simplicity, we assume each observed, noise corrupted principal submatrix is SPSD so that the eigendecompositions in steps 1 and 2a of the algorithm are well defined. In the noiseless case, to guarantee the uniqueness of $\hat{A}$, $A4$ required each $A(\iota_l, \iota_l)$ to be of rank r. In the noisy case, we place a similar requirement on $\tilde{A}(\iota_l, \iota_l)$, where we recognize that the rank of each $\tilde{A}(\iota_l, \iota_l)$ may be larger than r due to noise.

(A5) There exists a collection $\Omega_1, \ldots, \Omega_k$ of subsets of Ω such that $A2$ holds and if $k > 1$, there exists an ordering $\tau_1, \ldots, \tau_k$ such that the rank of $\tilde{A}(\iota_l, \iota_l)$ is greater than or equal to r for each $l \in \{2, \ldots, k\}$.

(A6) There exists a collection $\Omega_1, \ldots, \Omega_k$ of subsets of Ω such that $A2$ holds and $\tilde{A}_l \in \mathcal{S}_+^{n_l}$ for each $l \in \{1, \ldots, k\}$.

In practice, any $\tilde{A}_l$ which is not SPSD can be decomposed into the sum of a symmetric and an antisymmetric matrix. The negative eigenvalues of the symmetric matrix can then be set to zero, rendering a SPSD matrix. As long as this resulting matrix meets the rank requirement in $A5$, it can be used in place of $\tilde{A}_l$. Our algorithm can then be used without modification to estimate $\hat{A}$.

Theorem 4. *Let Ω index an $n \times n$ matrix $\tilde{A}$ which is a perturbed version of the rank r matrix A such that $A1 - A6$ simultaneously hold for a collection of principal submatrices indexed by $\Omega_1, \ldots, \Omega_k$.*

*Let $b \geq \max_{l \in [k]} \|C_l\|_F$ for some $C_l \in \mathbb{R}^{n_l \times r}$ such that $A_l = C_l C_l^T$, $\zeta \geq \lambda_{l,1}$ and $\delta \leq \min\{\min_{i \in [r-1],} |\lambda_{l,i} - \lambda_{l,i+1}|, \lambda_{l,r}\}$ for all l. Assume $\|A_l - \tilde{A}_l\|_F \leq \epsilon$ for all l for some $\epsilon < \min\{b^2/r, \delta/2, 1\}$. Then if in step 2 of Algorithm 1, **rank**$\left\{\hat{C}_l(\phi_l, :)^T \hat{C}(\iota_l, :)\right\} = r$ for all $l \geq 2$, Algorithm 1 will estimate an $\hat{A}$ from the set of principal submatrices of $\tilde{A}$ indexed by Ω such that*

$$\left\|A - \hat{A}\right\|_F \leq 2G^{k-1}L\|C\|_F\sqrt{r\epsilon} + G^{2k-2}L^2 r\epsilon,$$

where $C \in \mathbb{R}^{n \times r}$ is some matrix such that $A = CC^T$, $G = 4 + 12/v$, and $v \leq \lambda_r(A(\iota_l, \iota_l))/b^2$ for all $l \geq 2$ and $L = \sqrt{1 + \frac{16\zeta}{\delta^2} + \frac{8\sqrt{2}\zeta^{1/2}}{\delta^{3/2}}}$.

The proof is left to the appendix and is accomplished in two parts. In the first part, we guarantee that the ordered eigenvalues and eigenvectors of each $\tilde{A}_l$, which are the basis for estimating each $\hat{C}_l$, will not be too far from those of the corresponding A_l. In the second part, we bound the amount of additional error that can be introduced by learning imperfect $\hat{W}$ matrices which result in slight misalignments as each $\hat{C}_l$ matrix is incorporated into the final estimate for the complete $\hat{C}$. This second part relies on a general perturbation bound for the Procrustes problem, derived as Lemma 30 in the appendix.

Our error bound is non-probabilistic and applies in the presence of adversarial noise. While we know of no existing results for the recovery of matrices from deterministic samplings of noise corrupted entries, we can compare our work to bounds obtained for various results applicable to random sampling schemes, (e.g., [29,30,83,84]). These results require either incoherence [29,83], boundedness [84] of the entries of the matrix to be recovered or assume the sampling scheme obeys the restricted isometry property [30]. Error is measured with various norms, but in all cases

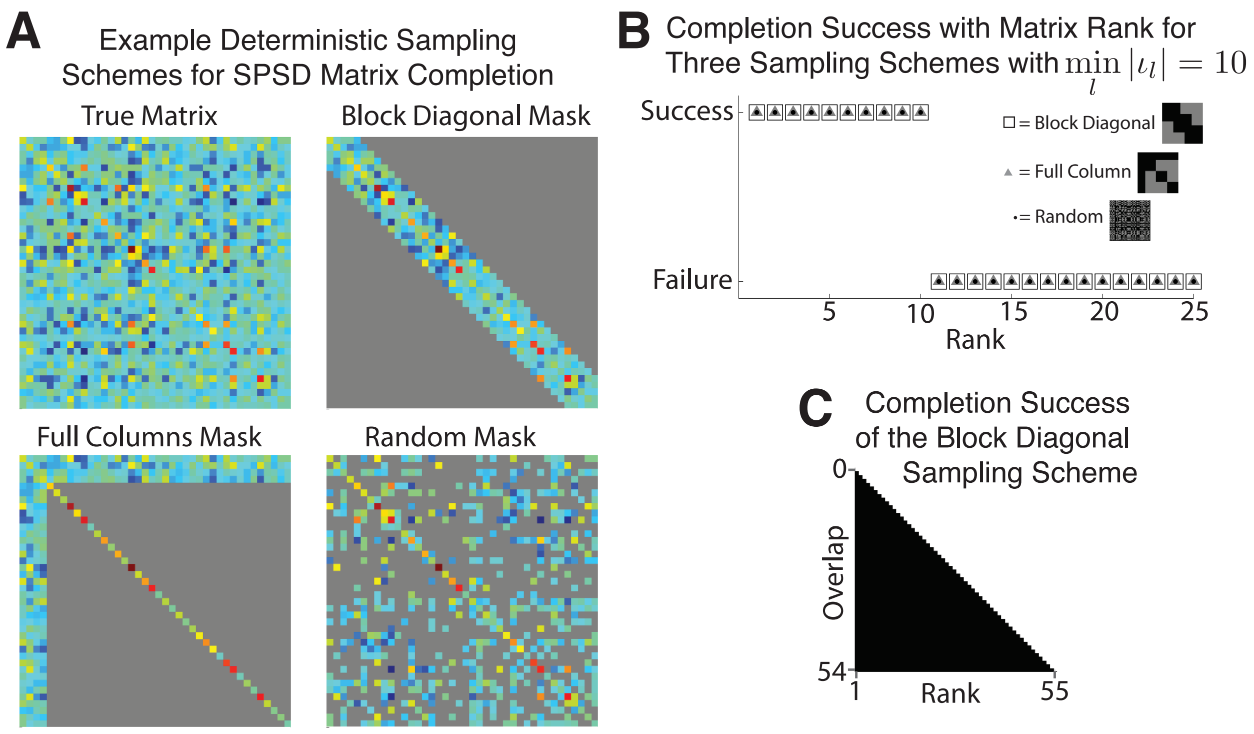


Figure 3.3: Noiseless simulation results. (A) Example masks for successful completion of a rank 4 matrix. (B) Completion success as rank is varied for masks with minimal overlap ($\min_l |\iota_l|$) of 10. (C) Completion success for rank $1 - 55$ matrices with block diagonal masks with minimal overlap ranging between $0 - 54$.

shows a linear dependence on the size of the original perturbation. For this initial analysis, our bound establishes that reconstruction error consistently goes to 0 with perturbation size, and we conjecture that with a refinement of our proof technique we can prove a linear dependence on ϵ. We provide initial evidence for this conjecture in the results below.

3.6 Simulations

We demonstrate our algorithm's performance on simulated data, starting with the noiseless setting in Fig. 3.3. Fig. 3.3A shows three sampling schemes, referred to as masks, that meet assumptions $A1$ through $A3$ for a randomly generated 40×40 rank 4 matrix. In all of the noiseless simulations, we simulate a rank r matrix $A \in \mathcal{S}_+^n$ by first randomly generating a $C \in \mathbb{R}^{n \times r}$ with entries individually drawn from a $\mathcal{N}(0,1)$ distribution and forming A as $A = CC^T$. The *block diagonal* mask is formed from 5×5 principal submatrices running down the diagonal, each principal submatrix overlapping the one to its upper left. Such a mask might be encountered in practice if we obtain pairwise measurements from small sets of variables sequentially. The l^{th} principal submatrix of the *full columns* mask is formed by sampling all pairs of entires, (i, j) indexed by $i, j \in \{1, 2, 3, 4, l + 4\}$ and might be encountered when obtaining pairwise measurements between sets of variables, where some small number of variables is present in all sets. The *random* mask is formed from principal submatrices randomly generated to conform to assumptions $A1$ through $A3$ and demonstrates that masks with non-obvious structure in the underlying principal submatrices can conform to assumptions $A1$ through $A3$. Algorithm 1 correctly recovers the true matrix from all three masks.

In panel Fig. 3.3B, we modify these three types of masks so each is made up of size 25×25 principal submatrices and $\min_l |\iota_l|$, the minimal overlap of a principal submatrix with those ordered before it, is 10 for each and attempt to reconstruct random matrices of size 55×55 and increasing rank. For matrices of rank $r \leq 15$, corollaries $7 - 9$ in the appendix can be applied to these scenarios to establish the necessity that $\min_l |\iota_l|$ be greater than or equal to r for a rank r matrix. We see that matrix recovery is successful for rank 10 or less, as predicted by theory, and unsuccessful for matrices of greater rank. In Fig. 3.3C, we show this is not unique to masks with minimal overlap of 10. Here we generate block diagonal masks with minimal overlap between the principal submatrices varying between 0 and 54. For each overlap value, we then attempt to recover matrices of rank 1 through $o + 1$, where o is the minimal overlap value. To guard against false positives, we randomly generated 10 matrices of a specified rank for each mask and only indicated success in black if matrix completion was successful in all cases. Matrix completion failed exactly when the rank of the underlying matrix exceeded the minimal overlap value of the mask. Identical results were obtained for the full column and random masks.

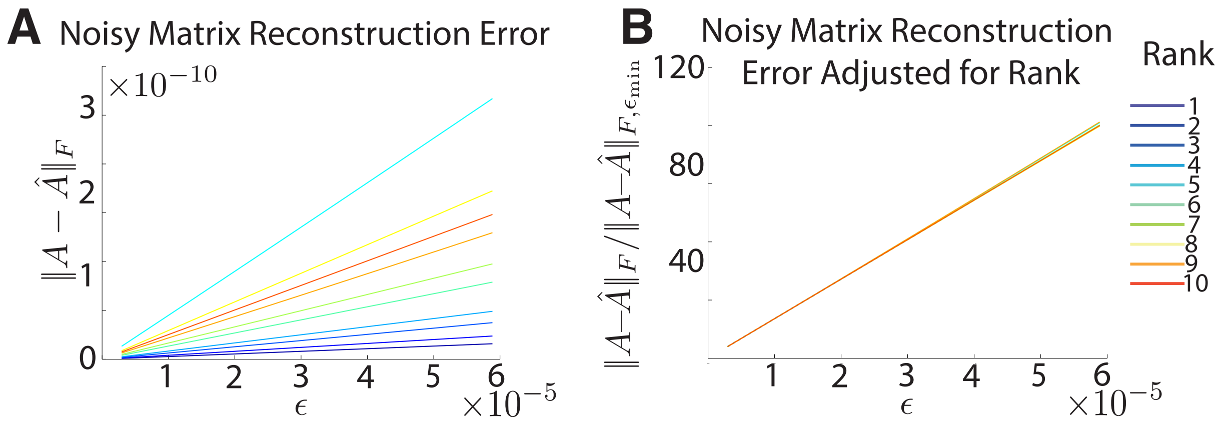


Figure 3.4: Noisy simulation results. (A) Reconstruction error with increasing amounts of noise applied to the original matrix. (B) Traces in panel (A), each divided by its value at $\epsilon = \epsilon_{\min}$.

We provide evidence the dependence on ϵ in Theorem 4 should be linear in Fig. 3.4. We generate random 55×55 matrices of rank 1 through 10. Matrices were generated as in the noiseless scenario and normalized to have a Frobenius norm of 1. We use a block diagonal mask with 25×25 blocks and an overlap of 15 and randomly generate SPSD noise, scaled so that $\|A_l - \tilde{A}_l\| = \epsilon$ for each principal submatrix. We sweep through a range of $\epsilon \in [\epsilon_{\min}, \epsilon_{\max}]$ for a $\epsilon_{\min} > 0$ and a $\epsilon_{\max}$ determined by the matrix with the tightest constraint on ϵ in theorem 4. Fig. 3.4A shows that reconstruction error generally increases with ϵ and the rank of the matrix to be recovered. To better visualize the dependence on ϵ, in Fig. 3.4B, we plot $\|A - \hat{A}\|_F / \|A - \hat{A}\|_{F,\epsilon_{\min}}$, where $\|A - \hat{A}\|_{F,\epsilon_{\min}}$ indicates the reconstruction error obtained with $\epsilon = \epsilon_{\min}$. All of the lines coincide, suggesting a linear dependence on ϵ.

3.7 Discussion

In this chapter we present an algorithm for the recovery of a SPSD matrix from a deterministic sampling of its principal submatrices. We establish sufficient conditions for our algorithm to exactly recover a SPSD matrix and present a set of necessity results demonstrating that our stated conditions can be quite useful for determining when matrix recovery is possible by any method. We also show that our algorithm recovers matrices obscured by noise with increasing fidelity as the magnitude of noise goes to zero. Our algorithm incorporates no tuning parameters and can be computationally light, as the majority of computations concern potentially small principal submatrices of the original matrix. Implementations of the algorithm, which estimate each $\hat{C}_l$ in parallel, are also easy to construct. Finally, our results could be generalized when the principal submatrices our method uses for reconstruction are themselves not fully observed. In this case, existing matrix recovery techniques can be used to estimate each complete underlying principal submatrix with some bounded error. Our algorithm can then reconstruct the full matrix from these estimated principal submatrices.

An open question is the computational complexity of finding a set of principal submatrices which satisfy conditions $A1$ through $A4$. However, in many practical situations there is an obvious set of principal submatrices and ordering which satisfy these conditions. For example, when recording from separate subpopulations of a larger neural population, all pairs of neurons recorded in one subpopulation would map to an observed principal submatrix of the covariance matrix for the entire population.

Chapter 4

Factor Analysis Models And Stitching

In this chapter we present a general method of combining neural population activity across space and time. We will follow Turaga et al. [22] and refer to this general problem as one of "stitching" neural recordings together. We begin the chapter by presenting the stitching problem at a high level. We then present a general probabilistic model, factor analysis (FA), which we apply to model neural activity with low-dimensional structure. We will show that given such a model for a population of neurons, the underlying low dimensional state (the expression level of the different activity patterns) can be inferred even when all of the neurons are not observed. Thus, even if different populations of neurons are recorded at different points in time, the low-dimensional state of the brain can be inferred and related among the different recordings.

Critical to the success of this approach is having knowledge of an FA model describing the joint activity of the union of all neurons that were recorded at different points in time. In practice, such a model must be learned from the individual recordings themselves. While a standard approach to fitting FA models, expectation maximization (EM), can be extended to the stitching problem, we will see that such an approach is lacking in two ways. First, practically, the straight forward extension of EM to the stitching scenario produces a fitting algorithm prone to local optima. Second, the derivation of adapted EM equations in itself tells us nothing about necessary conditions for successfully learning an FA model in the stitching scenario. Both of these short comings can be addressed by the application of the matrix completion theory just presented in chapter 3.

4.1 The Stitching Scenario

In a stitching scenario, we assume we would like to describe the joint activity of a large population of neurons but can only record from subsets of neurons from the whole population at any given time. Fig. 4.1 diagrams two such scenarios at a high level. The first scenario, illustrated in Fig. 4.1A, could arise when an experimenter is able to move her electrodes or region she is optically imaging to record from sequentially overlapping populations of neurons during an experimental session. This would be useful when trying to increase the total number of neurons sampled in an experiment. Such a sampling structure might also arise in recordings over long periods of time, where due to recording instability neurons appear and disappear over the course of an experiment. The goal of stitching would then be to relate the activity of the overlapping populations throughout the whole experiment.

The second recording scenario, illustrated in Fig. 4.1B, could arise when the experimenter is able to simultaneously record from a large population of neurons in one area and only a small number of neurons simultaneously in a second area. For example, a researcher may be able to place a multielectrode array in visual area V1 which can record from tens to hundreds of neurons at once and a movable single electrode in the middle temporal visual area (MT), which typically records from single neurons. By moving the electrode during the experiment, a larger number of neurons in MT could be sampled. Such an approach could be useful when searching for dimensions in the neural activity in which populations of neurons in the two areas communicate [10, 96].

These two examples are not exhaustive but serve as concrete examples of stitching scenarios. With these providing intuition, we now define stitching for the purpose of this work as 1) learning an FA model for the joint activity of a population of neurons when all members of the modeled population were not recorded simultaneously and 2) given such a model, inferring the underlying low-dimensional state at a given point in time from the observed activity of a subset of the modeled neurons. In the results that we present in chapter 5 and chapter 6, we will show the potential this holds for BCI, for studying learning and for studying the neural basis of communication. In this chapter we present the basic methods and theory.

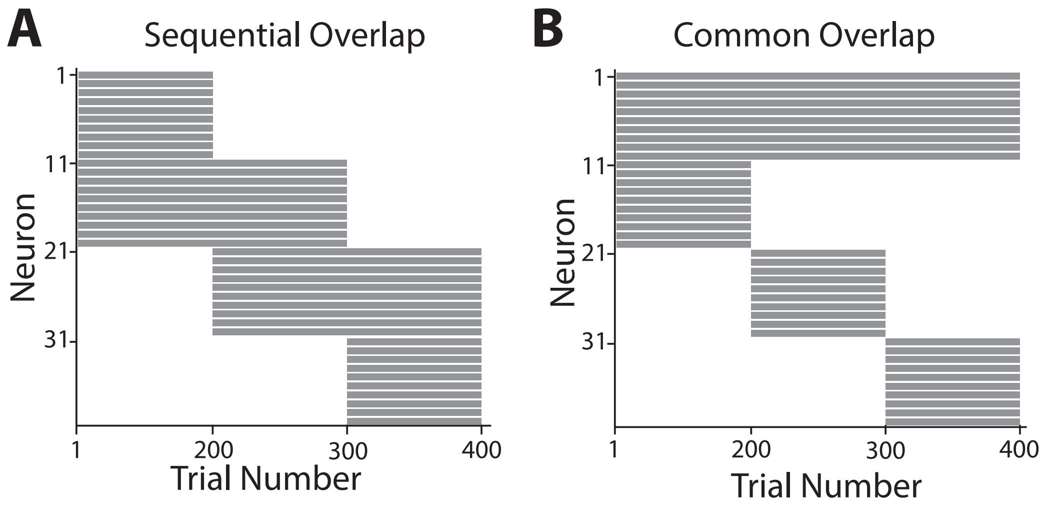


Figure 4.1: Two examples of recording from a population of neurons in blocks. (A) Three groups of 20 neurons are recorded for 200 trials each. Recordings are done sequentially so each group has 10 neurons in common with the group(s) before and/or after it. (B) Three groups of 20 neurons are again recorded for 200 trials each. Now, however, the same set of neurons, neurons 1-10, are shared in common among all the groups.

4.2 Factor Analysis Models of Neural Population Activity

We now describe a generative probabilistic model of neural population activity with low dimensional structure. Up to this point in the thesis, we have defined neural population activity to be low dimensional if its activity at any point in time can be described by the expression level of a small number of basis firing rate patterns for the entire population. Such a description is adequate when considering trial averaged neural activity, when we wish to describe the state of a population of neurons at different points in time during an average trial. This is because averaged across many trials, the individual Poisson like spiking noise of neurons is averaged out, revealing the underlying low dimensional structure in the firing rates of the population. We will refer to the individual spiking noise of a neuron as it's *private noise*.

For single trial analyses, the private noise of neurons obscures low dimensional structure in the underlying firing rates of the population. Techniques which can account for the private noise of neurons to reveal the underlying low dimensional structure of neural activity on a moment by moment basis are thus required. Factor analysis (FA) [97–99] is a common technique for doing just this.

We will continue to measure the activity level of neurons with spike counts. Under an FA model a low dimensional neural state, $l_t \in \mathbb{R}^m$, is related to the high dimensional population vector of spike counts $y_t \in \mathbb{R}^n$ through a linear transformation $C \in \mathbb{R}^{n \times r}$. Formally, we write the model as

$$y_t = Cl_t + \mu + \epsilon_t \tag{4.1}$$

$$l_t \sim \mathcal{N}(0, I) \tag{4.2}$$

$$\epsilon_t \sim \mathcal{N}(0, \Psi), \tag{4.3}$$

where $\mu \in \mathbb{R}^n$ is a vector of mean spike counts, $\epsilon_t \in \mathbb{R}^n$ captures spiking noise, Ψ is a $n \times n$ diagonal matrix capturing the variance of each neuron and I is the $m \times m$ identity matrix.

Having defined a model, we can now give precise definitions to the terms we have been using in an informal manner up to this point to describe low dimensional activity. Specifically, the columns of C contain the basis patterns which are combined to describe the neural activity, y_t, at time t. These patterns are combined in a linear way, by weighting each with its appropriate entry in l_t and summing the result. In other words, the vector l_t captures what we have been informally referring to as the "expression level" of each of the neural patterns at time t. We will subsequently simply refer to the vector l_t as the low-dimensional neural state at time t.

Given the low-dimensional state at time t, under the FA model, the activity of an entire population of neurons is determined up to independent spiking noise, the variance of which is captured for each neuron in Ψ. This is an important concept for the remainder of the thesis as it implies that if we know the low-dimensional state of a population of neurons, we know the expected behavior of the entire population, even if we can't observe each member of that population at all times.

4.3 Inferring Low Dimensional Neural State in the Stitching Scenario

We use probabilistic methods to estimate the low dimensional state of a neural population given a vector of observed neural activity. If all the neurons in an FA model are observed, the low-dimensional neural state, l_t, given a vector of observed spike counts y_t is normally distributed, $\mathcal{N}(\hat{l}_{l_t|y_t}, \Sigma_{l_t|y_t})$ with

$$\hat{l}_{l_t|y_t} = C^T B^{-1}(y_t - \mu) \tag{4.4}$$

$$\Sigma_{l_t|y_t} = I - C^T B^{-1} C, \tag{4.5}$$

where $B = (CC^T + \Psi)$. If all the neurons in y_t are observed, the maximum-a-posteriori (MAP) estimate for l_t is then $\hat{l}_{l_t|y_t}$.

In a stitching scenario, in general, the spiking activity for the entire population of neurons will rarely be observed. Instead, we will observe a subset of neurons at any given time. Allow ρ_t to index the set of neurons observed a time t. Continuing with the convention of using MATLAB notation, at time t, we then observe $z_t = y_t(\rho_t)$. We can form a joint probabilistic model l_t and z_t by marginalizing out the unobserved neurons not indexed by ρ_t from the FA model. Due to the linear, Gaussian nature of the FA model this can be done analytically. The posterior distribution for l_t given z_t is then again normally distributed $\mathcal{N}(\hat{l}_{l_t|z_t}, \Sigma_{l_t|z_t})$, where

$$\hat{l}_{l_t|z_t} = C(\rho_t, :) {B_t}^{-1}(z_t - \mu(\rho_t)) \tag{4.6}$$

$$\Sigma_{l_t|z_t} = I - C(\rho_t, :)^T B_t'^{-1} C(\rho_t, :), \tag{4.7}$$

where $B_t' = C(\rho_t, :)C(\rho_t, :)^T + \Psi(\rho_t, \rho_t)$. The MAP estimate for l_t when stitching is then $\hat{l}_{l_t|z_t}$.

Comparison of eqs. 4.4 and 4.6 and eqs. 4.5 and 4.7 reveals the satisfying interpretation that posterior distributions for l_t can be formed just as in the non-stitching scenario by simply using only the portions of C, Ψ and μ that pertain to the neurons observed at time t. The derivations for eqs. 4.6 and 4.7 can be found in appendix C.

4.4 Learning the FA Model Parameters for Stitching

We now turn to the important question of fitting an FA model in the stitching scenario. Expectation maximization (EM) [75] is a common method of fitting FA models when the set of spike count vectors for the model $y = \{y_1, \ldots, y_t\}$ are fully observed. Let Θ denote the set of parameters of an FA model, that is Θ is composed of C, Ψ and μ. Then the EM algorithm attempts to maximize the likelihood of the training data, $P(y; \Theta)$ with respect to Θ.

EM is an iterative procedure for finding model parameters which maximize the likelihood of training data. Iteration i of the algorithm produces updated estimates of model parameters and is composed of two steps. We will refer to the estimate for iteration i as Θ_i. In the expectation (E) step, the posterior distribution of the unobserved variables in the model, in this case the set of low-dimensional states $\ell = \{l_1, \ldots, y_t\}$ conditioned on y, $P(\ell | y; \Theta_{i-1})$ is calculated using the model parameters estimated in the previous iteration of the algorithm. In the maximization (M) step, new parameter estimates are produced by maximizing the expected log-likelihood of the joint observed and unobserved variables, $\Theta_i = \mathrm{argmax}_\Theta \, \mathbb{E}\left(P(y, \ell; \Theta)\right)$, where the expectation is taken with respect to the distribution over ℓ found in the E-Step. EM is known to converge to a local optima of the likelihood.

In the neural stitching scenario, we extend the EM algorithm by maximizing the log-likelihood of the observed data. That is we form the set $z = \{z_1, \ldots, z_t\}$ and attempt to maximize $P(z; \Theta)$. Derivations for the EM algorithm under the stitching scenario are given in the appendix.

4.4.1 EM and Local Optima

While the EM algorithm is generalized to the neural stitching scenario, empirically we observe that its susceptibility to finding local optima is greatly increased in the presence of structured, missing data. A typical result of initializing the EM algorithm randomly for simulated data is shown below in Fig. 4.1a-c. In this simulation, we select an FA model with 5 latent dimensions and 40 observed variables and draw data from it in three sequential blocks of 20 variables each. In each block, we record 1000 trials, and each block overlaps the other blocks by 10 variables, in a manner similar to Fig. 4.1a. In Fig. 4.2a, we plot the true value of CC^T, which captures the covariance structure among individual variables imposed by their joint relationship to the underlying latent factors. In Fig. 4.2b, we plot $\hat{C}\hat{C}^T$, which is the covariance structure, calculated from the $\hat{C}$ found by the EM algorithm when randomly initialized. Fig. 4.2c shows $CC^T - \hat{C}\hat{C}^T$. The EM algorithm settles on a estimate for $\hat{C}$ that incorrectly captures the covariance structure between neurons recorded in different blocks. This observation motivates us to develop a novel method of initializing the EM algorithm below.

4.4.2 Matrix Completion to Initialize EM

Motivated by the observation that the EM algorithm randomly initialized tends to return suboptimal parameter estimates, we propose a novel method of initializing the EM algorithm for the neural stitching scenario. As shown in the appendix, in the stitching scenario, the MLE estimate for μ can be calculated in a straight forward way without EM. However, analytical methods for finding the MLE estimates for C and Ψ do not currently exist. EM is an alternative, but it is not obvious how to initialize EM with respect to these parameters. Observe, however, that the covariance matrix, Σ, for the observations of the FA model can be written as

$$\Sigma = CC^T + \Psi. \tag{4.8}$$

Inspection of 4.8 reveals that Σ will be the sum of a low rank matrix, CC^T, and a diagonal matrix, Ψ. If we had a method of learning this decomposition, we could use an eigendecomposition to estimate C. We could then use these parameter estimates as starting values to initialize the EM algorithm with.

Decomposing Σ into CC^T and Ψ is made difficult for two reasons. First, in a stitching scenario with finite data, we will only observe a set of noisy principal submatrices of Σ and not the entirety of Σ. Second, the addition of Ψ to CC^T renders Σ full rank, so we cannot directly apply our theory for SPSD matrix completion.

Fortunately, we can extend our work on low-rank SPSD matrix completion to this scenario and learn the decomposition through a two step procedure. In the first step, a separate FA model is fit to the data from each set of simultaneously recorded neurons. Indexing the blocks of recordings from 1 to K, this returns model parameters $\hat{C}_1, \ldots, \hat{C}_K$ and $\hat{\Psi}_1, \ldots, \hat{\Psi}_K$ for each block. As each FA model is fit to a block of data with no missing entries, the EM algorithm is less prone to local optima. Each $\hat{C}_l$ matrix provides an orthogonally transformed and noisy estimate of corresponding rows of the full C matrix. As with Algorithm 1, we can then form a full $\hat{C}$ matrix by aligning and combining the $\hat{C}_l$ matrices returned from each EM model. We can also learn a single $\hat{\Psi}$ from the appropriate portions of $\hat{\Psi}_1, \ldots, \hat{\Psi}_k$.

We refer to this method of initializing the EM algorithm as the method of "rotations," as fusing the separate $\hat{C}_i$ matrices into a single $\hat{C}$ matrix can be understood as aligning each $\hat{C}_i$ to a consistent basis. Fig. 4.2d-e shows

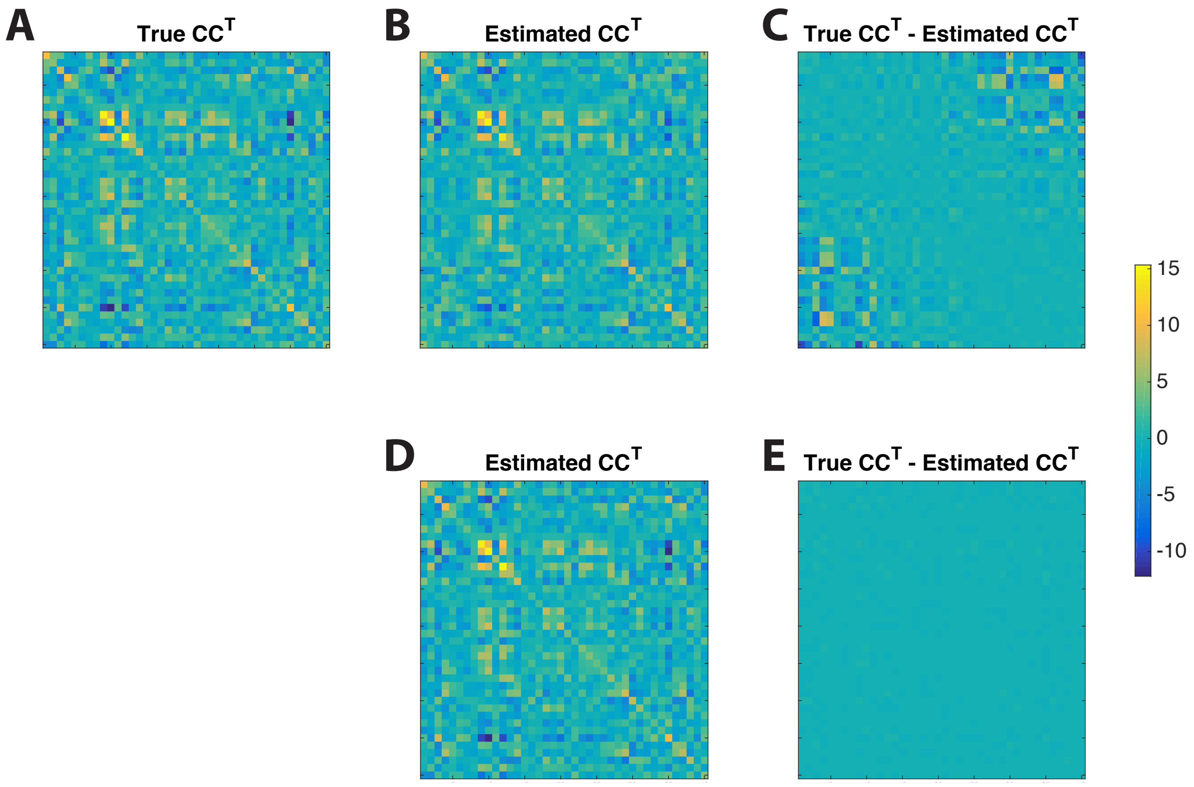


Figure 4.2: Fitting an FA model to simulated data when the true underlying parameters are known. (A) The true covariance structure, CC^T, from the underlying FA model. (B) The covariance structure, $\hat{C}\hat{C}^T$, estimated from EM randomly initialized. (C) The difference between CC^T and $\hat{C}\hat{C}^T$ estimated from EM randomly initialized. (D) The covariance structure, $\hat{C}\hat{C}^T$, estimated from EM initialized with the method of rotations. (E) The difference between CC^T and $\hat{C}\hat{C}^T$ estimated from EM initialized with the method of rotations.

the final parameter estimates when EM is run on the same data as in Fig. 4.2a-c, but initialized with the method of rotations. In this example, EM initialized with the method of rotations returned better parameter estimates than EM initialized with random parameters.

We now compare the average performance of the EM algorithm initialized randomly and with the method of rotations. We simulate sampling 1000 trials from each of three groups of 40 variables each, again similar to Fig. 4.1a, but vary the amount of overlap between the groups. For each overlap value, we generate 100 FA models with 10 latent variables, and use EM to learn model parameters. For each model, we initialize EM randomly and with the method of rotations. We can then compare the log-likelihood of the training data under EM initialized in both ways as well as the accuracy of the estimated covariance structures, $\hat{C}\hat{C}^T$. To guard against outliers, we check to ensure EM converged for both methods and throw out any simulations where EM initialized randomly or by the method of rotations did not converge in an allotted number of iterations.

The results of this analysis are shown in Fig. 4.3. In Fig. 4.3a the average difference in the log-likelihood of the model parameters on the training data is shown when the EM algorithm is initialized with the method of rotations and randomly. For all overlap values, initialization with the method of rotations is no worse than that of random initialization. Specifically, as overlap increases, the method of rotations more robustly learns a fused $\hat{C}$ matrix, allowing the final performance of the EM algorithm initialized with this method to increasingly outperform that of the EM randomly initialized. For large overlap values, which correspond to less missing data, the EM

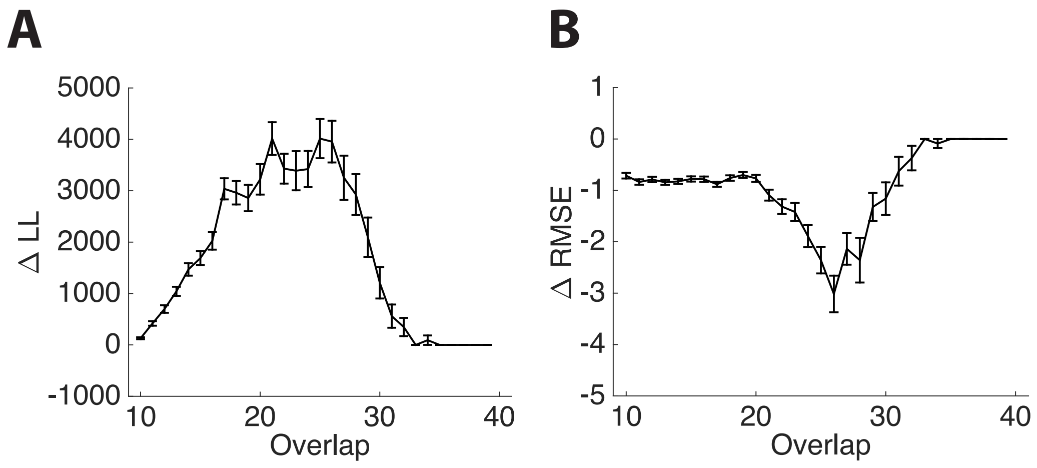


Figure 4.3: Relative performance of the EM algorithm for simulated stitching datasets (see text for details). (A) Average value of the log-likelihood of the training data when EM is initialized with the method of rotations minus that of EM initialized randomly. (B) Average value of the root mean squared error of $\hat{C}\hat{C}^T$ estimated when EM is initialized with the method of rotations minus that of EM initialized randomly. Error bars represent standard error in both panels.

algorithm becomes less susceptible to local optima when randomly initialized, and as expected EM converges to parameter estimates with equivalent log-likelihood values under both initialization methods. This same trend is reflected Fig. 4.3b when comparing the accuracy of the covariance structures estimated by EM initialized in both ways.

4.5 Necessary Conditions for Neural Stitching

We now turn to the important question of identifying necessary conditions for stitching to succeed. We begin by introducing notation and definitions we will need for this task. As much as possible, we have tried to keep the notation logically consistent with that used in chapter 3.

The problem we consider is one of learning an FA model for a total of M neurons. We assume that subpopulations of these neurons are recorded in K blocks. We will refer to such a scenario as a K-block stitching scenario. In each block, all neurons in the block are recorded simultaneously. Note that for brevity we will refer to recording from individual neurons, but everything that we will derive here also applies to channels of multiunit activity, LFP, etc. Let $\rho_k \subseteq \{1, \ldots, M\}$ be the set of indices for the neurons recorded in block k. We assume we record for a total of T trials, indexed by $1, \ldots, T$, and for block k we will let $\nu_k \subseteq \{1, \ldots, T\}$ be the set of trials for block k. Thus, $K, T, \rho_1, \ldots, \rho_K$ and $\nu_1, \ldots, \nu_K$ characterize a stitching scenario.

We now state a series of definitions important for our remaining theoretical work. We begin formally defining the overlap of a block with the other blocks in a recording.

Definition 5. *The* overlap *of block k, denoted ι_k, is defined as* $\iota_k = \rho_k \cap \left(\cup_{j=1}^{k-1} \rho_j \right)$.

In words, Definition 5 defines the overlap of block k to be a subset of the neurons recorded in the block such that any neuron in the overlapping subset has been recorded in at least one previous block. This may seem to be a counter intuitive means of defining overlap, as we only consider neurons indexed in blocks before block k as those with potential to be in the overlapping set. However, we will see that we can still state useful necessary conditions with overlap defined in this way.

We will state necessary conditions for learning FA models of the form presented in eqs. 4.1-4.3. With respect to these equations, we will use the notation $\Theta = \{\mu, \Psi, C\}$ for the set of parameters for an FA model. We now introduce the notion of equivalent parameter sets for an FA model.

Definition 6. *Let $\Theta = \{\mu, \Psi, C\}$ and $\Theta' = \{\mu', \Psi', C'\}$ be two sets of parameters for FA models as described eqs. 4.1-4.3. We say these parameter sets are* equivalent *if and only if*

1. $\mu = \mu'$
2. $\Psi = \Psi'$
3. $CW = C'$,

for some $W : WW^T = I$.

In words, we say two parameter sets are equivalent if they have identical means and private variances and the loading matrices can be related to one another through an orthogonal transformation. This transformation can be thought of as a changing the coordinate system in which latent states are expressed. More formally, it can be shown that under the FA model considered in this work, a vector, y, of observed neural data is distributed as $\mathcal{N}(\mu, \Sigma)$, with $\Sigma = CC^T + \Psi$. Notice that for any orthogonal W,

$$\Sigma = CC^T + \Psi = CWW^TC + \Psi = C'C' + \Psi.$$

Therefore, the likelihood of observed data is the same for two FA models with equivalent parameter sets, rendering the two unidentifiable.

We will now build up the concepts we will need to speak of identifiability in the stitching setting. Consider the random variable $\mathcal{Y} = \{Y_1, \ldots, Y_T\}$, where each Y_t is generated according to an FA model. We use capital letter notation here to denote that we are working with random variables and not their realizations. One can think of $\mathcal{Y}$ as a random variable for the neural activity for an entire experiment when we have the luxury of recording from every neuron in our population of interest at all times. We now define a new random variable for a stitching experiment, $\mathcal{Z}$. Given the sets of neurons we record from in each block of a stitching scenario, $\rho_1, \ldots, \rho_K$, and the trial indices in each of these blocks, $\nu_1, \ldots, \nu_K$, we define $\mathcal{Z} = \{Z_1, \ldots, Z_T\}$, where

$$Z_i = Y_i(\rho_k) : i \in \nu_k. \tag{4.9}$$

In words, $\mathcal{Z}$ is the random variable formed from $\mathcal{Y}$ by restricting the set of observed neurons to only those neurons we record from on each trial. We can now state our final definition related to the identifiability of a stitching model. Let z be a realization of $\mathcal{Z}$.

Definition 7. *Consider an FA model parameterized by Θ and the stitching scenario characterized by $\rho_1, \ldots, \rho_K$ and $\nu_1, \ldots, \nu_K$. We say the FA model is recoverable under this stitching scenario if*

$$P(z|\Theta) \neq P(z|\Theta') \tag{4.10}$$

for all Θ' not equivalent to Θ.

Under Definition 7, given a stitching scenario, a FA model is unrecoverable if there are two non-equivalent sets of parameters that result in the same probability distribution for $\mathcal{Z}$. We now introduce one final definition we will need to state our necessity results in crisp terms.

Definition 8. *The loading matrix, $C \in \mathbb{R}^{n \times r}$, of an FA model is said to have full rank submatrices if every submatrix of C with full columns and r or more rows is full column rank.*

Definition 8 has two benefits. First, in practice the true loading matrix for an FA model will not be known before collecting data, and Definition 8 will allow us to state our necessity results in a general way. Notice that if the entries of C are independently randomly generated from many common distributions (e.g., Gaussian), with high probability C will have full rank submatrices. Second, we will be building on the necessity results introduced in section 3.4. These necessity results hold for all but a set of degenerate matrices. By positing that the loading matrix for an FA model has full rank submatrices, we will be able to ignore such degenerate matrices when stating the results below.

With these preliminaries out of the way, we can now state our first necessity result.

Theorem 9. *Consider a two block stitching scenario for an FA model with a loading matrix, $C \in \mathbb{R}^{n \times r}$, which has full rank submatrices. Assume $|\rho_1| \geq r$, $|\rho_2 \setminus \iota_2| \geq r$ and $\rho_1 \cup \rho_2 = [n]$. If $|\iota_2| < r$, the model is not recoverable.*

Theorem 9 applies to a stitching scenario consisting of just two blocks of recordings when we wish to fit an FA model with a latent dimensionality of r. In the first block we record at least r neurons simultaneously. In the second, we continue to record from some number of neurons from block one and we also record from at least r new neurons. Theorem 9 allows us to conclude that we will not be able to recover the true FA parameters for our model unless we have at least r overlapping neurons between the blocks.

Below, we state two more theorems appropriate for stitching scenarios with more than two blocks. However, we first pause here to present the proof of Theorem 9 to illustrate the connection between SPSD matrix completion and the stitching problem.

Proof. We will prove that under the conditions of Theorem 9, there exist at least two non-equivalent parameter sets Θ and Θ' such that $P(z|\Theta) = P(z|\Theta')$. We start by noting that we can decompose $P(z|\Theta)$ as

$$P(z|\Theta) = \left[\prod_{t \in \nu_1} P(z_t|\Theta)\right]\left[\prod_{t \in \nu_2} P(z_t|\Theta)\right]. \tag{4.11}$$

By writing the joint density for an FA model parameterized by $\Theta = \{\mu, C, \Psi\}$ and marginalizing it can be shown that for $t \in \nu_k$

$$Z_t \sim \mathcal{N}\left(\mu(\rho_k), C(\rho_k, :)C(\rho_k, :)^T + \Psi(\rho_k, \rho_k)\right). \tag{4.12}$$

Now consider the matrix $M = CC^T$ with submatrices $M(\rho_1, \rho_1) = C(\rho_1, :)C(\rho_1, :)^T$ and $M(\rho_2, \rho_2) = C(\rho_2, :)C(\rho_2, :)^T$. C has full rank submatrices so $\textbf{rank}\{C\} = r$ and therefore it must be that $\textbf{rank}\{M\} = r$. By the same reasoning and the assumptions that $|\rho_1| \geq r$ and $|\rho_2 \setminus \iota_2| \geq r$ it follows that $\textbf{rank}\{M_1\} = r$ and $\textbf{rank}\{M(\rho_2 \setminus \iota_2, \rho_2 \setminus \iota)\} = r$.

We now make an appeal to matrix completion theory by considering the problem of observing only $M(\rho_1, \rho_1)$ and $M(\rho_2, \rho_2)$ and trying to recover the full matrix, M. By assumption $|\iota_2| < r$, and Lemma 20 in the appendix can be applied to concluded that there is no unique rank r completion in this scenario.

However, if there is no unique rank r completion, this implies there exists a C and C' which are not orthogonal transforms of one another such that $M(\rho_1, \rho_1) = C(\rho_1, :)C(\rho_1, :)^T = C'(\rho_1, :)C'(\rho_1, :)^T$ and $M(\rho_2, \rho_2) = C(\rho_2, :)C(\rho_2, :)^T = C'(\rho_2, :)C'(\rho_2, :)^T$. Define $\Theta' = \{\mu, C', \Psi\}$. By eqs. 4.11 and 4.12, it then follows that $P(Z|\Theta) = P(Z|\Theta')$. Further, since C and C' are not related through an orthogonal transformation, Θ and Θ' are non-equivalent parameter sets and this completes the proof.

□

The statement of Theorem 9 raises a natural question. It seems the requirement of r non-overlapping neurons in block two is somewhat strong. Would the necessity of having r overlapping neurons between the blocks still stand if the number of non-overlapping neurons in block two was reduced? While we do not present a formal proof, for practical purposes the answer is most likely yes. The intuition is that what makes an FA model unlearnable is an inability to learn the correct alignment of the different portions of the loading matrix estimated from each recording. Consider the extreme example of recording two populations of neurons with no overlap. If the portion of the loading matrix for one of these populations is uniformly 0, then any alignment of the two estimated portions of the loading matrix will suffice. While we do not present a proof, we conjecture that under randomly generated loading matrices, the set of loading matrices which would fall into such a special category is measure zero. Thus, in a real stitching scenario, it seems there is little chance relaxing the requirement on the number of non-overlapping neurons in block two would change the necessary conditions for stitching.

We now state our next theorem, which is appropriate when we wish to record from a large number of neurons in a sequence of overlapping blocks. We state the formal theorem and then below discuss in more concrete terms the situations it applies to and its implications. Its proof can be found in appendix C and as with the proof for Theorem 9, it relies heavily upon the necessity results we have proven for the SPSD matrix completion problem.

Theorem 10. *Consider a K-block stitching scenario for an FA model with a loadings matrix, $C \in \mathbb{R}^{n \times r}$, which has full rank submatirces. Assume $|\rho_1| \geq r$, $|\rho_i \setminus \iota_i| \geq r$ for all $i \geq 2$ and $\cup_{k=1}^{K} \rho_k = [n]$. Finally, assume that $|\rho_i \cap \rho_j| = \emptyset$ if $|i - j| > 1$. Then the model is not recoverable if there exists an $i \geq 2$ such that $|\iota_i| < r$.*

In words, Theorem 10 considers the following stitching scenario:

1. In the first block of trials, we record from at least r neurons simultaneously.
2. In each subsequent blocks, we record from at least r neurons that we've never recorded from before. Additionally, we drop neurons from blocks so that two non-sequential blocks have no neurons in common.

If these conditions are met, Theorem 10 allows us to conclude that the number of overlapping neurons between blocks must be at least r for an FA model with r latent dimensions to be recoverable.

We now state our final theorem which is appropriate when we can stably record from a large population of neurons while at the same time recording from a second population of neurons in a sequential manner. As noted above, such a scenario would be useful if we can record a large population in one area (e.g., V1) and a small, changeable population in another area (e.g., MT).

Theorem 11. *Consider a K-block stitching scenario for an FA model with a loadings matrix, $C \in \mathbb{R}^{n \times r}$, which has full rank submatirces. Assume $|\rho_1| \geq r$, $|\rho_i \setminus \iota_i| \geq r$ for all $i \geq 2$ and $\cup_{k=1}^{K} \rho_k = [n]$. Finally, assume $(\rho_i \setminus \rho_1) \cap (\rho_j \setminus \rho_1) = \emptyset$ for any pair $(i, j) \in \{(i, j) \in [k] \times [k] | i \neq 1, j \neq 1, i \neq j\}$. Then the model is not recoverable if there exists an $i \geq 2$ such that $|\iota_i| < r$.*

In words, Theorem 11 considers the following stitching scenario:

1. In the first block of trials, we record from at least r neurons simultaneously.
2. In each subsequent block, we record from at least r neurons that we've never recorded from before, while possibly also recording from some of the neurons that we recorded in block 1.
3. If any two blocks have any neurons in common, these must have been recorded in block 1.

If these conditions are met, Theorem 11 allows us to again conclude that the number of overlapping neurons between blocks must be at least r for an FA model with r latent dimensions to be recoverable.

Chapter 5

Stitching for Self-Recalibrating BCI Regression

5.1 Background

As discussed in the introduction and in chapter 2, due to instability in the signals recorded from modern multi-electrode arrays, current intracortical BCIs must be retrained daily. In chapter 2, we presented a self-recalibrating classifier, capable of recalibrating itself during normal daily use. This could potentially free future patients from the burden of enduring dedicated recalibration sessions. In this chapter, we now consider the problem of designing a self-recalibrating BCI decoder for regression.

Unlike the the case for classification, others have proposed self-recalibrating intracortical BCI decoders for regression [38–41]. Most of these proposed methods (e.g., [38, 40, 41]) use a form of self-training, which is a method of semi-supervised learning that retrains a decoder on unlabelled data, using the decoders output in place of ground truth labels [100]. Such an approach has been shown to work online. Indeed, Li et al. [41] demonstrate a self-training procedure that maintains stable BCI performance in a closed loop setting over 11 separate BCI sessions. However, it is foreseeable that decode performance with self-training procedures could degrade over time as trials with poor estimates of user intent will be used to retrain the decoder as if the estimates were ground truth.

In this chapter we present a self-recalibrating decoder for regression that requires no self-training. We do so by leveraging the low-dimensional structure of neural signals, which should be consistent from day-to-day. Recent work by Sadtler et al. [12] has provided evidence that the low dimensional structure of neural activity in motor cortex is more than merely descriptive but can reveal causal constraints on the neural activity patterns a population of neurons can exhibit. This leads us to conceptualize the problem of a subject controlling a BCI as one of modulating not individual neurons but the underlying neural state. The logic is that in reality a subject does not learn to modulate the activity of only those neurons a BCI electrode array records. Instead, a subject must modulate an entire network of neurons, only some of which are recorded by the BCI array.

To decode from low-dimensional neural state, we introduce a two-stage decoder, which first uses FA to estimate low-dimensional state and then uses a Kalman filter to estimate a user's intent from this inferred state. We use an unsupervised method of identifying stable electrodes between days, allowing us to apply stitching to update the parameters of the FA portion of the decoder from day-to-day. The key insight is that as long as the FA model is appropriately updated from day-to-day, which can be done using only recorded neural activity, the decoder can be automatically recalibrated without the need to resort to self-training.

5.2 Methods

5.2.1 Notation and Binning of Neural Data

On a given day, we will decode threshold crossings from each of M electrodes on a multielectrode array. That is instead of assigning spikes to putative neural units, we group all voltage events crossing a set threshold together for each electrode and simply count the number of these threshold crossing events on each electrode. We note, however, that the algorithm we describe here can be extended to allow for the use of sorted units as well.

Unlike classification where we observed a single bin of spikes per trial, in regression we observe a series of non-overlapping spike counts, $y_1, \ldots, y_T$, where T is the total number of bins we observe for a particular trial and $y_t \in \mathbb{N}^M$, where M is the number of electrodes we record from. In this work, we use 45 millisecond bin sizes. Each bin of spike counts, y_t, is associated with a low dimensional neural state, $l_t \in \mathbb{R}^r$, where r is the dimensionality of the latent state and the user's intended two dimensional cursor velocity $v_t \in \mathbb{R}^2$.

5.2.2 Neural Recordings

In this study, we make use of data collected in collaboration with the lab of Aaron Batista at the University of Pittsburgh. All animal handling procedures were approved by the University of Pittsburgh Institutional Animal Care and Use Committee. A rhesus macaque was trained to perform a standard center out BCI cursor control task to 8 targets using the two component decoder described in section 5.2.3. Unlike most BCI experiments, the parameters of the decoder (described below) were learned on a session of data not analyzed here (session 0) and then held fixed across 12 sessions of data, each occurring on a separate day, which we analyze here.

Each session consisted of a set of observation and brain-control trials. In this work, we analyze only the observation trials. In an observation trial, the monkey's neural activity was not used to control the cursor. Instead, the cursor was automatically moved from a center target to a peripheral target as the animal watched. Neural activity in motor cortex has been shown to modulate when an animal observes but does not perform a task under these conditions [101–104]. Each trial consisted of a 300 millisecond freeze period in which the peripheral target was visible but the cursor did not move followed by a 900 millisecond period in which the cursor moved straight towards the target. In each session, there are 10 observation trials to each target, for a total of 80 observation trials per day.

Use of observation trials allows us to avoid the confounds that would be present if the animal was modulating his neural activity in a closed loop setting in a manner specific to the decoder used in the closed loop experiments. This is important because we seek to apply a new decoder, our self-recalibrating decoder, to these data, and the offline performance of the self-recalibrating decoder could be confounded if the prerecorded neural signals we use for evaluation contain online corrective signals.

5.2.3 A Two-Stage Decoder for BCI Regression

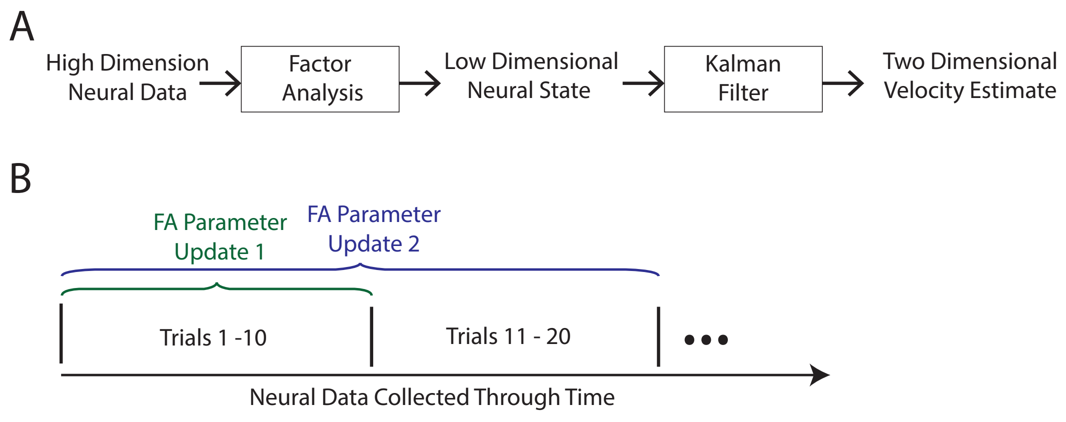

A
High Dimension Neural Data
Factor Analysis
Low Dimensional Neural State
Kalman Filter
Two Dimensional Velocity Estimate
B
FA Parameter Update 1
FA Parameter Update 2
Trials 1 -10
Trials 11 - 20
•••
Neural Data Collected Through Time

Figure 5.1: (A) A diagram of the two-stage decoder used in this work. In the first stage, a FA model is used to infer low-dimensional state. This low-dimensional state is then fed to a Kalman filter which estimates intended velocity. (B) The base two-stage decoder is made self-recalibrating by using stitching to refit the parameters of the FA model throughout a trial. Here we illustrate how neural data is accumulated in blocks of 10 trials. After each block of trials, the accumulated data is used to refit the FA parameters.

To develop a self-recalibrating decoder, we modify a two-stage decoder introduced by Sadtler et al. [12] in a study of neural constraints for learning. While the work of Sadtler et al. focused on BCI as a scientific tool, we seek to leverage the underlying decoder design to produce a better decoder for use in the clinic. The basic concept is illustrated in Fig. 5.1. The decoder is made up of two components. The first component employs a FA model to infer low-dimensional brain state from high-dimensional neural activity. The second component is a Kalman filter which estimates user intent, in this case the desired two dimensional velocity of a computer cursor, from the inferred low-dimensional brain state.

The Kalman filter has been used as a decoder in multiple BCI studies (e.g., [36,105,106]). However, it is typically used to map directly from observed neural activity to user intent. The novelty of our design is the introduction of FA to first infer low-dimensional brain state. In an offline setting with sufficient training data, a single component decoder which uses a Kalman filter to map directly from high-dimensional neural activity to estimated velocity should perform just as well if not better than the two component decoder presented here. This is because the use of FA filters out information from any neural signals which lay outside of the subspace determined by the loading matrix of the FA model.

However, in an online setting, the introduction of FA allows us to use stitching to render the decoder self-recalibrating. This is because the parameters of an FA model can be learned from the neural data alone, while the parameters of the Kalman filter remain fixed, producing a self-recalibrating decoder.

Formal description of the two-stage BCI decoder

We now formally present the two-stage decoder. For now, we present the decoder for a fixed set of parameters. In section 5.2.4 we describe a method for automatically updating these throughout a decoding session.

The first stage of the decoder estimates low-dimensional state, l_t, from high-dimensional neural activity, y_t. We use the same FA model presented in chapter 4 to capture the relationship between l_t and y_t. Continuing with the notation introduced in chapter 4, we model:

$$\begin{aligned} l_t &\sim \mathcal{N}(0, I) \\ y_t | l_t &\sim \mathcal{N}\left(C l_t + \mu, \Psi\right). \end{aligned}$$

In this chapter, the dimensionality of l_t is always 10. From this model, given y_t, the maximum a posteriori estimate of latent state, $\hat{l}_t$, is formed as

$$\hat{l}_t = C^T (CC^T + \Psi)^{-1} (y_t - \mu). \tag{5.1}$$

In the second stage of the decoder, we estimate velocity v_t from estimated latent state, $\hat{l}_t$, with a Kalman filter. The latent dynamical system on which the Kalman filter is based on models

$$v_1 \sim \mathcal{N}\left(\pi_1, S_1\right) \tag{5.2}$$

$$v_t | v_{t-1} \sim \mathcal{N}\left(A v_{t-1}, Q\right), \tag{5.3}$$

for some mean vector $\pi_1 \in \mathbb{R}^2$ and covariance matrix $S_1 \in \mathcal{S}^2_+$ for the initial desired velocity, as well as matrix $A \in \mathbb{R}^{2\times 2}$ and covariance matrix $Q \in \mathcal{S}^2_+$ for the transition model. The Kalman filter then models the relationship between desired velocity and low-dimensional brain state as

$$l_t | v_t \sim \mathcal{N}\left(H v_t + g, R\right), \tag{5.4}$$

for $H \in \mathbb{R}^{r\times 2}$, $g \in \mathbb{R}^l$ and covariance matrix $R \in \mathcal{S}^l_+$.

The Kalman filter is used to predict estimated velocity from $\hat{l}_t$. Let K be the steady-state Kalman gain, which is the value of the standard Kalman gain as the number of steps in the Kalman filter goes to infinity. It can calculated form the Kalman filter parameters above. Then if $\hat{v}_t$ is the estimated desired velocity at time point t in a trial we estimate

$$\hat{v}_t = K(\hat{l}_t - g) + (I - KH)A\hat{v}_{t-1}. \tag{5.5}$$

For the first bin of a trial, the parameter estimate for π_0 is used in place of $\hat{v}_{t-1}$.

5.2.4 Stitching for Self-Recalibration

We now describe a procedure for automatically updating the FA parameters of the two-stage decoder to make it self-recalibrating. Note that prior to this, the parameters of the decoder for both the FA model and Kalman filter need to be learned once in a supervised manner. In our work, this was done on day 0, prior to any of the sessions analyzed here. See appendix D for more details. For the remainder of this work, we focus on adapting the parameters of the decoder in a self-recalibrating manner once initial values are learned.

The key intuition is that a single FA model can be learned in an unsupervised manner from neural data alone without knowing a user's intended velocity commands. This means that neural activity collected during normal BCI use can be used to fit updated FA models. However, the loading matrices for these models, as discussed in section 4.5, are only defined up to an orthogonal transformation. If this is not addressed, the coordinate systems for the latent variables of two models may be incongruent. Fortunately, we can use stitching to align the loading matrices for an FA model learned from data on later days to the loading matrix learned on session 0.

More specifically, we apply stitching to this problem by assuming that a subset of electrodes on the array maintain steady signal characteristics from day 0 to day k. Below we will describe a proposed method to identify this set of electrodes, but for now, we will take it as a given that we can identify this set. Index electrodes of the array from 1 to M and let ι denote the set of electrodes that we identify with consistent signal characteristics from day 0 to day k. Let C_0 and C_k be the loading matrices for FA models fit to neural activity recorded on the array on day 0 and day k, respectively. If the electrodes in ι maintain stable signal characteristics across days, we assume that their relationship to underlying low-dimensional state is preserved across days. Following the same reasoning as in section 4.4.2, this implies that an orthogonal matrix, W, can be learned such that $C_0(\iota, :) = C_k(\iota, :)W$. Once this W is identified, it can be used to rotate the whole of C_k into a coordinate system that is consistent with the FA model used for the day 0 decode. The necessity conditions derived in section 4.5 can be applied here to conclude that in practice we will need to identify at least as many stable electrodes as the dimensionality of the FA model in the decoder for this procedure to succeed.

Having described how an FA model can be learned on day k and aligned to a consistent coordinate system with the day one decoder, two questions remain. First, when during a trial should neural data be collected to learn the parameters of the FA model? To fit consistent models from day to day, we would like neural data to be collected when the subject is nominally engaged in the same task in a consistent manner. To account for this we, we choose to use vectors of spike counts binned in 45 millisecond non-overlapping bins between 405 and 900 milliseconds from the start of each trial. Analysis of data from the observation task described above indicates that this period is sufficiently delayed to allow for reaction and visual processing delays and not too near the end of a trial that the monkey has learned he can disengage and still achieve success.

The second question is how many trials of data should we wait to observe before updating the parameters of the FA model on day k? If we use too little data, we may estimate the parameters of the FA decoder poorly. If we use too much data, the user may have to endure a long period of suboptimal BCI use before we update the BCI parameters. In this initial work, we select a procedure which aims to update FA parameters once there is minimally sufficient data to reliably learn the FA parameters and which then iteratively relearns these parameters as more data is collected. Practically, this means we wait for ten trials before updating parameters and continue to update parameters in intervals of ten trials, fitting FA models to all of the accumulated data that has been observed during a session. This is illustrated in Fig. 5.1B. We find this works well on the data presented here, though alternative intervals can be investigated in future work.

Finally, we note that in this work that we always align the loading matrix fit during session k to the loading matrix originally fit to data from session 0. This is similar to the strategy we use in the self-recalibrating classifier where each decoding day begins with the same prior distribution over tuning parameters - no matter how the tuning parameters changed the day before. In the classification setting, this decision is made because tuning parameters seem to drift in a constrained manner from a fixed point, nominally captured by the prior. In the regression setting, we have not yet analyzed drift in the electrode signals as deeply, but we follow the findings for the case of classification and chose to align loading matrices for session k to the session 0 loading matrix. In future work, alternative schemes can be investigated.

5.2.5 Identifying Stable Electrodes Across Days

Each time we fit an FA model on day k, we must identify electrodes with stable neural signals between day 0 and day k. We identify stable electrodes across time by exploiting the low-dimensional covariance structure among electrodes. Intuitively, if an electrode becomes unstable, it's covariance with all the other electrodes on an array should change. We exploit this to rank electrodes by stability from day-to-day. After doing so, we select the most stable electrodes to stitch with.

More specifically, let C_0 and C_k capture the loading matrices for the FA model originally fit on day 0 and the FA model fit to data observed on day k for an array with M electrodes. For the sake of illustration, assume that neural signals on all electrodes have maintained a constant relationship to the underlying low-dimensional state. In this case, $L_0 = C_0 C_0^T$ will be equal to $L_k = C_k C_k^T$. Now, consider a scenario where the first electrode on the array changes its relationship to the underling latent state on day k. In this case we will still have

$$L_0(2:M, 2:M) = C_0(2:M, :)C_0(2:M)^T = C_k(2:M, :)C_k(2:M)^T = L_k(2:M, 2:M).$$

However, the row/column of L_k capturing the covariance structure of electrode one with all the other electrodes on the array, that is $L_k(1, 1:M)$ and $L_k(1:M, 1)$, will change. The matrix $D = L_k - L_0$ will capture this. Let $b \in \mathbb{R}^M$ be a vector such that $b(i) = \sum_{j=1}^{M} |D(i,j)|$. That is $b(i)$ is the sum of absolute value of the elements in the i^{th} row of D. Ranking electrodes by the their value in b, we will pick the electrode with the largest value as the most likely candidate to have changed. This procedure can be generalized to the cases of picking p electrodes that have changed by selecting electrodes with the top p values in b, and this is the procedure we use in this work for identifying unstable electrodes across days.

The question arises of how to pick a value for p? On data from stable recordings over multiple days, we find (see results below) that using a value of $p = 0$ is optimal. For stable recordings this is not surprising as all electrodes give useful information for learning an orthogonal transformation between C_0 and C_k. However, on this stable data, we find that stitching performance drops off only mildly as p is increased for values of p that are not too large. We therefore suggest picking a value of p that is large enough to contain a reasonable number of electrodes that are unstable but not so large that the identified set of stable electrodes is too small to allow for robust stitching. Below we systematically vary p and find that for an array with 91 active electrodes a value of $p \approx 30$ works well.

5.2.6 Supervised Recalibrating of the Two-Stage Decoder

While the aim of this work is to develop a self-recalibrating decoder, we also seek an estimate of "best case" decode performance. To approximate this, we will retrain the two-stage decoder described above in a supervised manner. We do this by breaking up each day's data into blocks of 10 trials. For a day with 80 observation trials, this results in 8 blocks. For each block of 10 trials that we decode, we train the decoder using the other 7 blocks of data. Notice that in this way the decoder is retrained in an acausal fashion, as trials that occur later in the day are used for retraining the decoder of trials that occur earlier. While this is impossible in practice, by maximizing the amount of training data available for retraining in each block, we hope to better approximate best case performance. We will refer to the decoder trained in this way as the "supervised-recalibrated" decoder.

When recalibrating the decoder in a supervised manner, just as with the the self-recalibrating procedure, we use neural data in bins between 405 and 900 milliseconds after the start of each trial. Unlike the self-recalibrating procedure, we also allow ourselves knowledge of the intended target. This allows us to take the decoded velocities in each of these bins and rotate them to point straight at the target. This is motivated by the assumption that at each point in time, a subject would desire to move the cursor straight toward the target. This general idea was proposed by Gilja et al. [36]. We note that even though we analyze observation trials, where the cursor is moved automatically from the center to peripheral target, the decoded velocities at each point will not point straight to the target. We use the rotation procedure to preserve the length of the decoded velocity vector but correct for errors in decoded direction.

After rotating velocity vectors in the selected bins they, along with the associated neural data, are used to retrain the decoder. To retrain the decoder, a new FA model is fit and low-dimensional states are inferred as described in appendix C. After this, the observation model of the Kalman filter relating desired velocity to low-dimensional state is fit in a supervised manner, using the rotated velocities as the intended velocities. We do not refit the parameters for the state transition model, as we do not expect these to change from day-to-day.

5.2.7 Simulated Data

In this work, we present two results. First, we run the non-recalibrated decoder used in the online experiments and our self-recalibrating decoder on the prerecorded data described above. This analysis indicates that in this particular 12 days of data, the neural signals recorded on all 12 days of data analyzed in this work appear to be fairly stable from day to day. While we can test our self-recalibrating decoder on this data, we seek a better test set of data where neural signals necessitate the use of a self-recalibrating decoder to maintain stable across-day performance. We turn to simulations to achieve this.

We perform the simulations in two parts. In the first part, we seek to simulate data with the same marginal statistics as those observed in the real data. We will do this by fitting parametric models to the data of a particular day. In the second part, we seek to simulate common array failure scenarios by changing the parameters of the models we use to generate the simulated data.

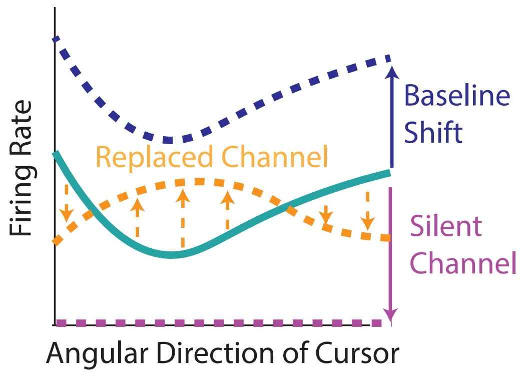


Figure 5.2: A conceptual diagram of how we simulate three failure scenarios. Each curve is a tuning curve relating the mean rate of threshold crossings on an electrode to angle of desired cursor movement. The solid, green curve illustrates the baseline tuning curve which is then perturbed. Silent channels are simulated by setting the mean rate (and standard deviation) of a channel to 0. Baseline shifts are simulated by applying a random offset to the entire tuning curve for an electrode. Tuning changes are simulated by switching the tuning curve of an electrode for that of another electrode. See text for more details.

Fitting Models to the Observed Data We seek a method of simulating data that does not a priori assume latent structure in the neural signals. Given one day's real data, we achieve this by fitting univariate Gaussians for each electrode at each decoded bin in a trial to each particular target. That is given electrode e, for bin t on a trial to target j, we calculate the mean $\mu_{e,j,t}$ and standard deviation, $\sigma_{e,j,t}$ of all the observed spike counts for that electrode bin and target combination observed in the real data. With 80 observation trials per day, this gives 10 data points with which to fit each Gaussian model.

Electrode Failure Scenarios There are many possible mechanisms by which the recorded signals on an electrode of an array could change from day-to-day. We do not seek to exhaustively simulate all possibilities. Instead we highlight three common mechanism of failure, illustrated conceptually in Fig. 5.2.

The first failure mechanism that we simulate is that an electrode fails to record any threshold crossings at all. This scenario is simulated by setting the mean and standard deviation to 0 for that electrode across all bins and targets. The second scenario we simulate is a change in baseline of an electrode, presumably due to changes in electrode impedance or micromovements of the array. This scenario is simulated by adding a constant value to the means for that electrode across all bins and targets. The final scenario we simulate is that the tuning characteristics of an electrode change due to detecting threshold crossings from the action potentials of different combinations of neurons in its vicinity from day to day. This would again presumably be attributable to micromovements of the array or changes in electrode impedance. To simulate this scenario, we chose to simulate decoding scenarios with less electrodes than were present in the real data. This allows us to completely replace the model parameters for an electrode with those of an electrode not initially selected for data simulation, an extreme form of tuning change. This may tend to over estimate changes in tuning curves but we note this will only serve to make it more difficult for the self-recalibrating decoder to retrain itself and so is conservative for our purposes.

Simulating Failure Scenario Data We now describe how we simulate decoding experiments. In this work, we select one day's data (day 1) to fit models for data simulation to as described above.

For each simulated dataset, we do the following. First, we randomly select 60 electrodes out of the original 91 to include in the decoder. After selecting these 60 electrodes, we marginalize the unused electrodes out of the FA model of the two-stage decoder that was used in the actual experiments. This then forms the base decoder for each day.

When simulating a failure scenario, we randomly select one electrode to become silent, 10 electrodes to undergo baseline shifts and 5 electrodes for tuning parameter changes. Electrodes are randomly selected so that a single electrode can be subject to one, two or three of these failure mechanisms. Baseline shifts for electrodes were randomly selected from a Normal distribution with a standard deviation of 20 Hz.

To simulate data for a trial to target j, a spike count for bin t and electrode e is drawn according to a $\mathcal{N}\left(\mu_{e,j,t}, \sigma_{e,j,t}\right)$ distribution, where $\mu_{e,j,t}$ and $\sigma_{e,j,t}$ will be modified from their values fit to the original data if electrode e has undergone a simulated failure. Randomly generated Gaussian values will not be non-negative integers, so generated values are rounded to the nearest whole number and floored at 0 to generate the simulated spike count.

For each simulated dataset we simulate 80 observation trials (10 trials to each target), and then decode the simulated data with the non-recalibrated, self-recalibrating and supervised-recalibrating decoders, each starting with FA models marginalized to the 60 electrodes originally selected for the decode scenario. For the results we present below, we simulate 100 stable datasets and 100 failure datasets randomly generated and decoded in this way.

5.3 Results

We begin our results by presenting the performance of the two-stage decoder when it is not recalibrated, self-recalibrated and recalibrated in a supervised manner as described above on all 12 sessions of prerecorded data. As illustrated in Fig. 5.3A, at each point in a trial, performance of the decoder was measured with angular error, which is the absolute value of the angle between a vector pointing straight from the cursor to the target and a decoded velocity vector. In this analysis, initial parameters of the decoder were learned in a supervised manner during session 0, prior to those sessions analyzed here. For the non-recalibrated decoder, we have kept these paraemeter values fixed across all 12 sessions of data. Fig. 5.3B presents average decode performance for the first 900 milliseconds of a trial for all 80 observation trials for each of the 12 sessions analyzed in this work. As can be seen in this figure, for the 12 days of data analyzed here, decode performance of the non-recalibrated decoder was fairly consistent from day-to-day. The high initial error at the start of each trial is attributable to reaction and visual delays. On multi-electrode arrays, particularly outside the period following immediate array implantation, it is not uncommon for neural signals to be relatively stable on the array over periods of weeks to months. This likely accounts for the stable performance of the non-recalibrated decoder on this dataset. However, this will not always be the case, and self-recalibrating algorithms will be important to maintain stability over longer periods of time and for arrays at different periods in their lifetime when neural signals are less stable from day to day.

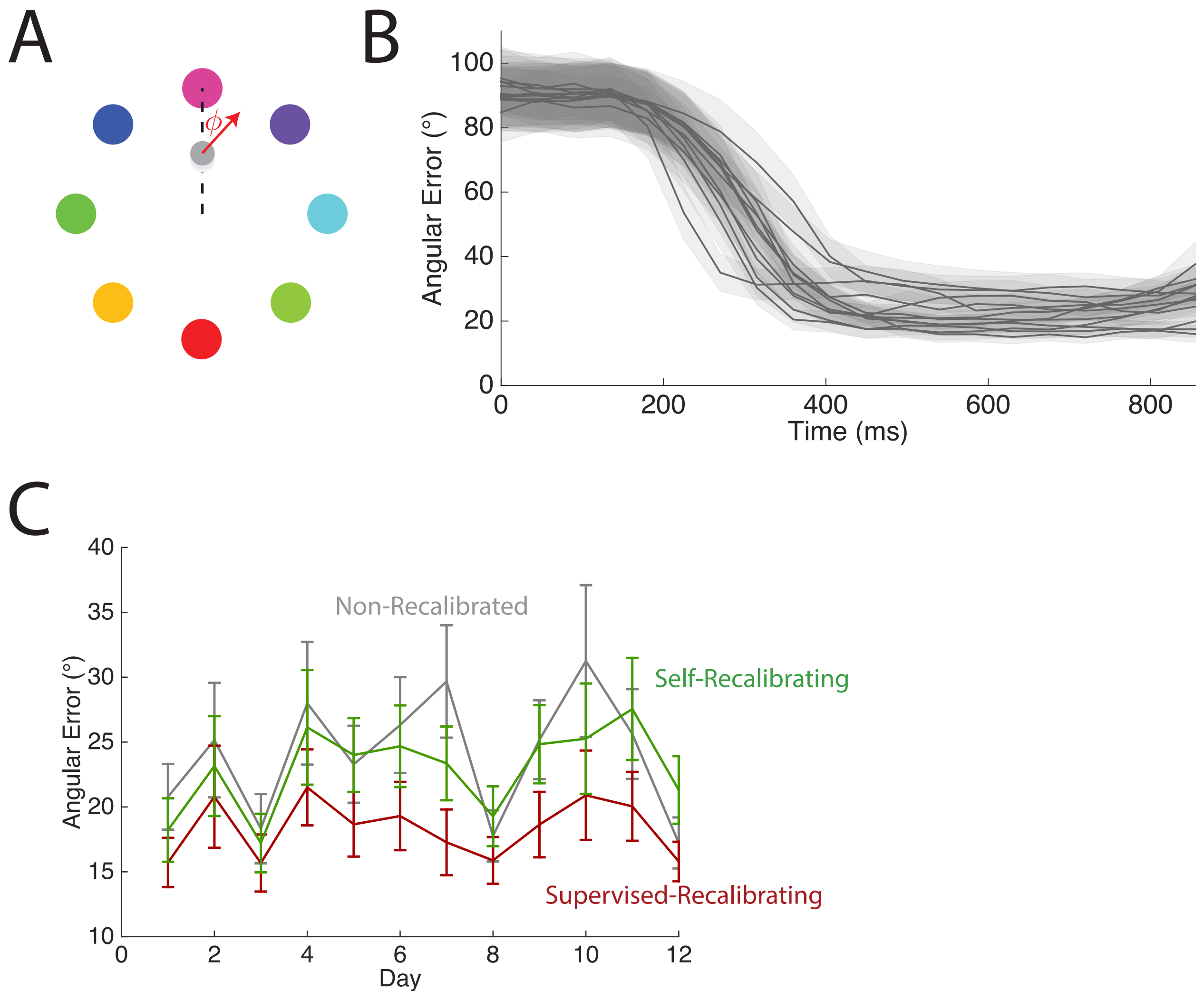


Figure 5.3: (A) Performance was measured as angular error. For each decoded bin of data, angular error was calculated as the unsigned angle between the direction of the decoded velocity and the direction pointing straight to the presented target. As we analyze only observation trials, irrespective of the decoded velocity, the cursor always moved along a preprogrammed trajectory straight from the center to the target. (B) Decode performance of the non-recalibrated decoder was relatively stable from day-to-day. Average angular error for all trials decoded in the first 900 milliseconds of a trial on each day are shown. Each line represents data for one day. Shading represents 95% confidence intervals. (C) Supervised retaining only modestly improves performance. Average error on each of the 12 days of analyzed data for bins decoded in the period between 405 and 900 milliseconds after trial start. Error bars represent 95% confidence intervals. Performance of non-recalibrated, the self-recalibrated and the supervised recalibrated decoders are shown.

In Fig. 5.3 we summarize the average angular error in bins in the window of 405 to 900 milliseconds after the start of a trial for all trials on each day for the non-recalibrated two-stage decoder, as well as the two-stage decoder self-recalibrated and recalibrated in a supervised manner, as described above. On the 12 days of data we analyze there are $M = 91$ active electrodes on the array. For the self-recalibrated results shown here, we have chosen p, the number of electrodes identified as unstable across days to be 31, so that we identify $91 - 31 = 60$ stable electrodes to stitch with. This value was chosen based on an analysis of the performance of the self-recalibrating decoder to differing values of p, described next. As can be seen, on most days the self-recalibrating decoder performs slightly better than the non-recalibrated decoder. We also compare against the two-stage decoder when it is retrained in a supervised manner as described in section 5.2.6. With supervised retraining the relative improvement in decode performance is modest, supporting the notion that neural signals were relatively stable from day to day. This modest improvement suggests a probable bound on the performance improvement we might expect to achieve with the self-recalibrating decoder.

We next evaluate the sensitivity of our self-recalibrating decoder when the method of identifying unstable electrodes described in section 5.2.5 is used with differing values of p. For each day, we sweep values of p between 81, corresponding to identifying only 10 stable channels across days and 0, corresponding to assuming all channels are stable across days. We then evaluate angular error in each 45 millisecond bin in the time range of 405 to 900 milliseconds from the start of each observation trial on each day. For each day and value of p, we average the values for all bins and trials to calculate the average angular error achieved. The results, showing the mean across all days for each value of p, are shown below in Fig. 5.4. As can be seen in the figure, performance initially improves sharply as more electrodes are identified as stable. After a certain point, performance continues to improve but in a more gradual manner. This gradual decline is consistent with stable neural signals on the 12 days of data analyzed here. In general, we will not know a priori if an array will be as stable from day-to-day as the array in this particular experiment. However, Fig. 5.4 shows evidence that, at least for this data, after a certain number of electrodes are identified as stable from day-to-day (≈ 30 in the figure), using more electrodes to stitch produces diminishing returns in performance. Armed with this observation, we suggest selecting p so that there are an ample number of electrodes to stitch with from day-to-day, but there is also a reasonable margin allowing for unstable electrodes to eventually appear.

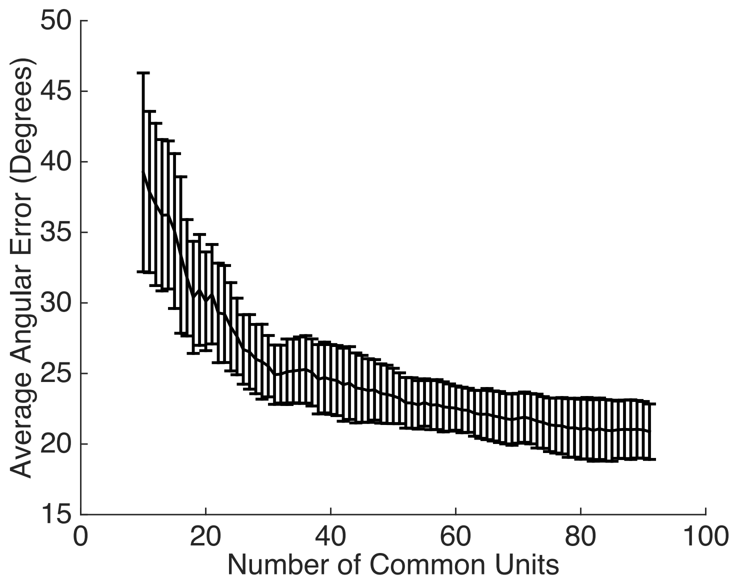


Figure 5.4: Average angular error across all 12 days of data of decoded as the number of units identified as common across days is swept from 10 to 91 for an array with 91 active electrodes. Angular error for a day was calculated for as the mean angular error across all trials for bins in the period between 405 and 900 milliseconds from the start of a trial. Error bars represent 95% confidence intervals.

Fig. 5.5 presents the results of the self-recalibrating decoder on simulated data. For these scenarios with arrays with 60 simulated electrodes, we chose a value of p to identify 40 stable electrodes across days. The value of 40 was chosen using the rule of thumb we suggest above: large enough so that FA models can be aligned between days reliably but small enough so that rows of the loading matrices for electrodes which are unstable are not used for alignment. Fig. 5.5A shows performance of the self-recalibrating, supervised-recalibrating and non-recalibrated decoders on data simulated as described in section 5.2.7 when we simulate no unstable electrodes. Similar to results on the real data, we see that in a scenario with stable electrodes from day to day, the self-recalibrating decoder mildly outperforms the non-recalibrated decoder. This is because the non-recalibrated decoder's parameters are fixed to those trained on day 0, while the simulated data is based on models fit to neural data from day 1. Therefore the models used to generate this data reflect drift in the neural signals that occurred between the two days, which is reflected in the simulated data. Fig. 5.5B shows performance of the three decoders when unstable electrodes are simulated as described in section 5.2.7. Performance of the non-recalibrating decoder is now degraded in comparison to it's performance on the stable data, while the self-recalibrating decoder is able to nearly match the performance it achieved under on the stable data.

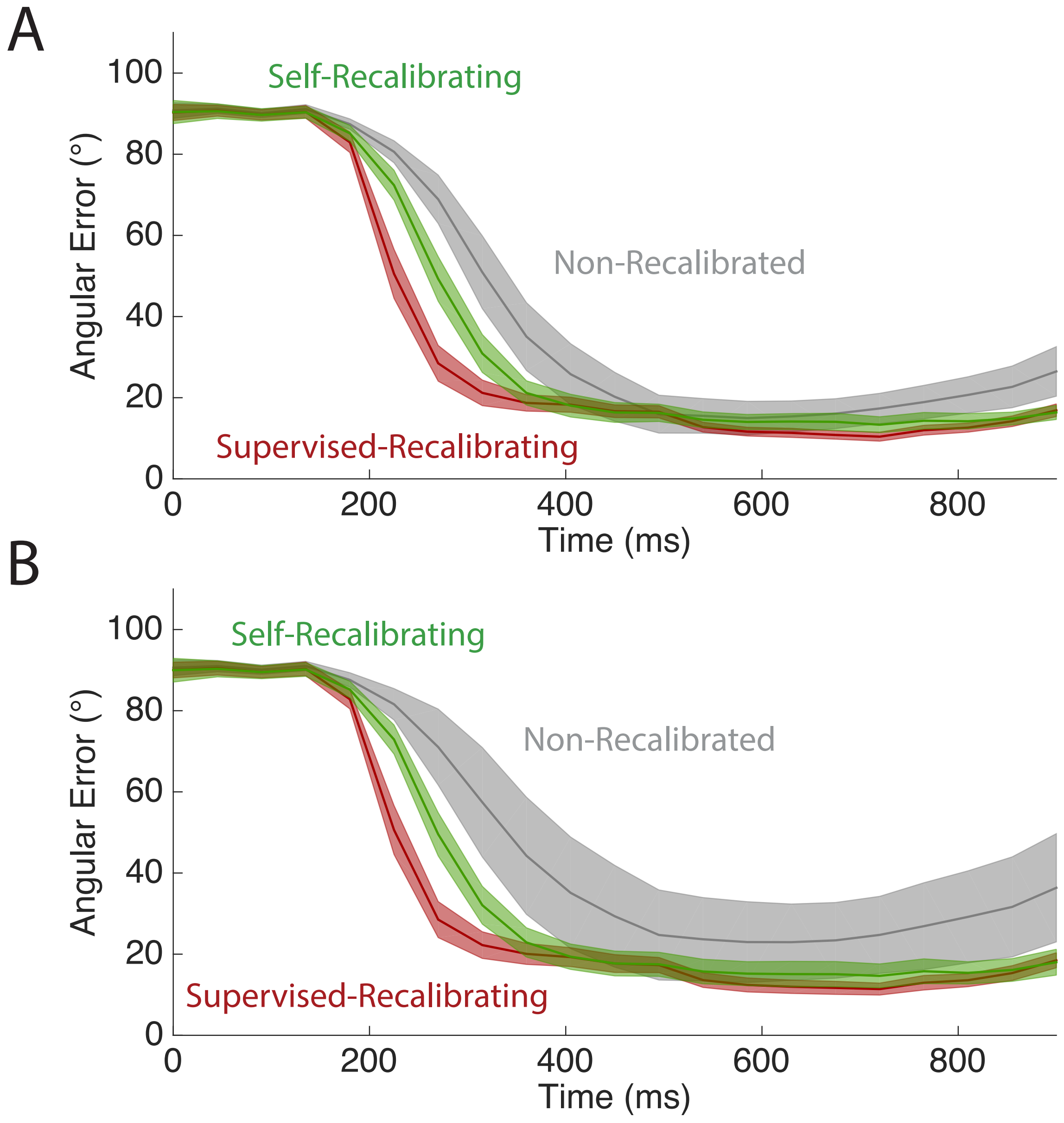


Figure 5.5: Average performance of the self-recalibrating, supervised-recalibrating and non-recalibrated decoders on simulated datasets without (A) simulated electrode failures and with (B). Solid lines represent the mean over 100 simulations, while shaded regions represent standard error.

Chapter 6

Stitching for Scientific Applications

We now turn from the use of stitching for self-recalibrating BCI and propose two ways in which stitching can be used for basic science. In the first half of this chapter we focus on the question of studying changes in neural activity patterns with learning. For long term learning experiments, we would like to study such changes across days, but this is made difficult if we cannot stably record from ensembles of neurons over many days. We propose to use stitching to address this problem. In the second half of the chapter, we consider the problem of studying interactions between neural populations in two different brain areas. Here the difficulty lies in recording large numbers of neurons in two areas simultaneously. We propose that stitching can be used to identify subspaces in which neural population activity in two areas covary, even if we cannot record simultaneously from all neural units in both populations of interest.

6.1 Stitching to Study the Neural Basis of Learning

In this section we propose stitching as a means of combining neural recordings across days to studying learning on long time scales. We will show that with stitching, even if it is impossible to record from every member of a population of neurons over multiple days, changes in the underlying low-dimensional state of the neural activity of the population can be tracked.

Multiple studies have considered the problem of studying neural population correlates of learning. Using single electrode recordings Paz et al. [107] track changes to the peristimulus time histograms (PSTH) of individual neurons in early and late learning in a motor adaptation task in primary motor cortex (M1) and supplementary motor area (SMA). They cluster the PSTHs of neurons, under the assumption each cluster reflects neurons with similar functional properties. They find an increase in the number of clusters in early adaptation in SMA and late adaptation in M1, which they interpret as signatures of dynamic reorganization of the neural populations. Durstewitz et al. [108] record from populations of neurons in prefrontal cortex during a within-day rule learning task. They visualized the progression of a low-dimensional representation of the neural state and fit hidden markov models (HMM) to their recorded data, finding evidence of abrupt transitions in neural activity that correlate with task success. By applying within-day perturbations to a BCI mapping, Sadtler et al. [12] showed that the low-dimensional structure present within neural population activity could be used as a basis for predicting the learnability of a BCI mapping. Finally, in a multiday learning experiment, Ganguly and Carmena [71] recorded from ensembles of 10 and 15 neurons in two monkeys performing a BCI task over 9 and 19 days and observe the emergence of stable maps between neural activity and the animal's behavior.

We seek to provide a computational method to enable multi-day learning studies as in Ganguly and Carmena. The ability to hold every member of the chosen ensembles over multiple days was critical to the success of this study. This is generally acknowledged to be difficult with modern electrode arrays. Ganguly and Carmena were aided by studying a task in which neural correlates of learning became manifest relatively quickly (days to weeks) and the use of relatively mature arrays for which they could identify relatively small sets of stable units. In this work, we propose that stitching can be used to study learning over longer periods, with large numbers of neural units in the face of instabilities in neural recordings. While the methods we present would be useful for addressing difficulties encountered with modern electrode arrays, they can be applied to a broad array of recording technologies where it may not be possible to record from all neurons of interest from day to day.

In the rest of this section, we describe how we can directly use the algorithm presented in chapter 4 to study learning over multiple days when the set of neurons we can record on any given day changes. Using data from the BCI learning task reported in [12], we then simulate a stitching scenario and demonstrate that we can track changes in neural state across a recording session. Finally, we conclude with additional simulations in which we quantify reconstruction error on estimated low-dimensional state as the rate of neuron turnover between recordings changes.

6.1.1 Methods

Stitching to Study Learning

The ideas presented in chapter 4 can be directly applied to study learning across days. The key idea is that changes in neural population activity in learning will be captured by changes in its low-dimensional state representation. Using stitching we can fit FA models and estimate the low-dimensional state from day to day, even if the makeup of the population of neurons we record from changes.

To do this, we apply the algorithm from chapter 4 directly. Continuing with the notation in that chapter, given neural population recordings from K days, $\rho_1, \ldots, \rho_K$ index the set of neural units recorded on each day. A single FA model is then fit to this data as described in section 4.4. After this single FA model is fit to the data, the underlying low-dimensional state can be estimated on any given day using eq. 4.6.

Thus, stitching for learning is a straightforward application of the algorithm presented in chapter 4. In this work, we assume that the overlapping neurons between days have been pre-identified. Techniques to do this are the subject of open research. In general, if sorted units are used, waveforms, interspike interval histograms, cross-correlograms, autocorrelograms and mean firing rate can all be employed to identify units across days [109, 110]. If neural activity is measured using threshold crossings on the electrodes of an array, the technique presented in section 5.2.5 can also be used.

Neural Recordings

We analyze data from experiments previously reported in [12]. Briefly, a rhesus macaque was trained to control a BCI for a center out task to 8 peripheral targets, similar to the task described in section 5.2.2. We analyze one experimental session of data (03-19-2012). In this session, after performing 406 trials with a decoder which was fit to the subject's neural activity (intuitive period), the parameters of the decoder were shuffled and the animal performed 874 trials with this new decoder (perturbation period). While we ultimately aim to apply stitching to across-day experiments, we start with an analysis of a within-day experiment as we assume signals on each electrode are stable across the duration of this experiment. From this base, we can then simulate stitching experiments where we assume we can only record from certain electrodes in blocks.

In the online experiment reported in [12], spikes were binned in 45 milliseconds non-overlapping bins. In the offline analysis we report here, we count threshold crossings on 86 electrodes during a 315 millisecond (7 bins) window starting 135 milliseconds (3 bins) after the bin in which the cursor was released from a freeze state in the online experiment. We include both successful and failed (trials in which the monkey failed to reach the target in an allotted time) trials in our analysis, as failed trials may still contain important neural correlates of learning, particularly before the animal has become proficient with the new mapping. There were a small number of trials which were completed before the end of the 315 millisecond window, and these were discarded from our analysis. This left a total of 387 trials in the intuitive period and 872 trials in the perturbation session for a total 1259 trials.

Evaluation with Simulated Stitching Scenarios

In the results we present below, we first present results on the prerecorded data meant to built intuition. The methods for that analysis can be stated in short form and are presented in line with the results below.

Here, we present the methods for the set of simulations we perform to quantify error in reconstructing low-dimensional state in different stitching scenarios. We seek to simulate data in a manner which reflects the statistics in the experimental data described above. We also chose to perform these simulations in such a way that we will have access to ground truth low-dimensional state. This will allow us to rigorously measure error of estimates for these states found in a stitching scenario.

Data Simulation Our simulations are accomplished in three steps:

1. In the first step, we fit a single statistical model, Gaussian Process Factor Analysis (GPFA) [9], to the set of spike counts collected across the $T = 1259$ analyzed trials in the prerecorded data described above. To fit the model, we break up the trials by target and period. This produces 16 time series consisting of a single spike count vector per trial for all trials to a given target within the intuitive or perturbation periods. We chose to employ GPFA as we expect that for reaches to a single target low-dimensional state should change smoothly through time with learning, and GPFA allows for the learning of the appropriate time constants governing the smoothness of neural dynamics. In the work we present below, we chose to fit GPFA with a low-dimensional state of 10 dimensions. After fitting this model, we infer the maximum a posteriori estimate for the underlying latent state for each trial. While this will be subject to estimation error and bias, we treat this as a reasonable approximation of how low-dimensional state progresses through time. We refer to the low-dimensional state for trial t estimated in this way as $l_t \in \mathbb{R}^k$, where k is the dimensionality of the GPFA model that we fit. We let the matrix $L \in \mathbb{R}^{T \times k}$ contain this set of latents in its rows.
2. In the second step of each simulation, we use the low-dimensional state estimated in step (1) to generate new, simulated spike counts for each trial. We do this by generating spike counts according to the parameters of the fit GPFA model. That is we generate a vector of neural activity, y_t for trial t as

$$y_t | l_t \sim \mathcal{N}\left(C_G l_t + \mu_G, \Psi_G\right),$$

   where C_G, μ_g and Ψ_G are the loading matrix, mean vector and matrix of private noises for the GPFA model fit in step (1). Notice that in this way, we have new spike counts generated from a model where we do know the ground truth latent state that generated any given spike count.
3. In the final step of each simulation, we generate stitching data by selecting only a given set of neurons to simulate recording from in each trial. We then fit an FA model to this data using the algorithm in chapter 4 and calculate the maximum a posteriori estimate of latent state, $l'_t \in \mathbb{R}^r$, for each trial, where $r \leq k$ is the dimensionality of the FA model we fit. Note that we allow r to differ from the latent dimensionality that was used to generate the data. Let the matrix $L' \in \mathbb{R}^{T \times r}$ contain this set of latent state estimates in its rows.

Quantifying Error Given a matrix of estimated latent states from a simulation, L', we describe how we measure its error with respect to the set of true latent states that were used to generate the simulated data contained in L. As r may be less k, we seek to transform the latents in L into a basis such that dimensions are ordered by amount of variance they explain in the neural data. This will allow us to compare the r dimensional latents in L' to the top r dimension in the transformed version of L. Denote the singular value decomposition of C_G as $C_G = U_G \Sigma_G V_G^T$. We transform the latents as $L_O = L V_G \Sigma_G$. L_O now contains the ground truth latents expressed with respect to a basis which orders the latent dimensions by the amount of variance of the neural data they explain under the GPFA model. We refer to transforming the latents in this way as orthonormalization [9].

We then measure the error of the latents in L' with respect to the ground truth latents in L_O. We note that the latents in L' will not in general be expressed with respect to the same basis as those in L_O. Therefore, we measure error after searching for a transformation of the basis for the latents in L' that produces the best error by solving

$$\mathcal{E} = \min_{A \in \mathbb{R}^{r \times r}} \frac{1}{\sqrt{T}} \left\| AL' - L_O(:, 1:r) \right\|_F, \tag{6.1}$$

where $L_O(1:r, :)$ are the top r ordered dimensions in L_O. In words, eq. 6.1 measures the root mean squared error between the latent states in L' and the top r dimensions of the latents states in L_O subject to finding the best coordinate transformation to align them. Note that eq. 6.1 explicitly searches over the set of all affine transformations.

This is important as the coordinate systems for the latent variables for FA models for different sets of neurons will not in general be relatable by an orthogonal transformation.

6.1.2 Results

We now present our results. Fig. 6.1A presents the monkey's performance over the course of the experiment. This plot was produced by convolving task success for each trial (0% for failure, 100% for success) with a Gaussian kernel with a standard deviation of 20 trials. Trials were smoothed separately during the intuitive and perturbation periods. As can be seen, the monkey was able to achieve near perfect success during the intuitive period. Performance was initially low at the start of the perturbation period, but increased rapidly in the first tens of trials of the perturbation period and then continued to increase gradually for the remainder of the perturbation session.

The performance trace in Fig. 6.1A is color coded to indicate 5 blocks of trials (2 in the intuitive period, 3 in the perturbation period). To build intuition, we estimate latent state for each trial during this experiment in two ways. In the first way, which we refer to as the simultaneous recording scenario, we use spike counts from a constant set of 80 of the 86 recorded electrodes on all trials to fit an FA model and infer low-dimensional state on each trial. In the second scenario, we again fit an FA model to spike counts from the same 80 electrodes, but we only reveal spike counts for 40 of the 80 electrodes on each trial. We do this in overlapping blocks, with overlap between blocks of 30 electrodes to mimic a stitching scenario. Panels B and D of Fig. 6.1 schematically depict which electrodes we allow ourselves access to spike counts from on each trial in both scenarios.

In panel D of Fig. 6.1 we present a two-dimensional projection of the orthonormlized low-dimensional neural state in both periods as recovered by fitting an FA model with a 5 dimensional latent state to the simultaneous scenario. The dimensions shown are the two which capture the most shared variance in the neural activity under the fit FA model. The inset of panel D shows a depiction of the 8 targets used in the task. For clarity, we do not show estimates of the low-dimensional state for individual trials in both periods. Instead, for the intuitive period, for trials to each target, we fit Gaussian distributions to the spread of estimated latents for each target and indicate regions containing 95% of the density of these distributions, color coded by target. For the perturbation period, we plot the trial-to-trial estimates of low-dimensional state for individual trials, again color coded by target. As can be seen, a low-dimensional state representation captures changes in neural activity patterns between the intuitive and perturbation sessions.

Panel E of Fig. 6.1 contains the estimates of latent state under the stitching scenario. To produce these results, we use the algorithm in chapter 4 to fit an FA model with a 5 dimensional latent state to the stitching scenario data and infer latents on each trial. Panel E is plotted in the same manner as panel D. The coordinate system of the 5 dimensional latents returned by the model have be transformed by using eq. 6.1 to learn a transformation that best aligns them to the 5 dimensional latents estimated under the simultanoues scenario to provide the best comparison. Comparison of panels D and E indicate that stitching recovers results which are qualitatively similar to those obtained in the simultaneous scenario. This is noteworthy for two reasons. First, the sets of electrodes used in the first and last blocks of the stitching scenario have no electrodes in common. Without stitching, it is not clear how a joint latent variable model could be fit to these groups of electrodes. Second, when inferring the underlying latent state for any given trial under the stitching scenario, spike counts for only 40 electrodes are available, as compared to 80 in the simultaneous scenario. Thus, for any trial, there is more information in the simultaneous scenario with which to estimate the latents. Despite this, at least in this two-dimensional projection, the estimated latents in the two scenarios appear remarkably similar. This is perhaps not surprising. For a latent state of a fixed dimensionality, as more electrodes are used for inference, we expect error to decrease asymptotically.

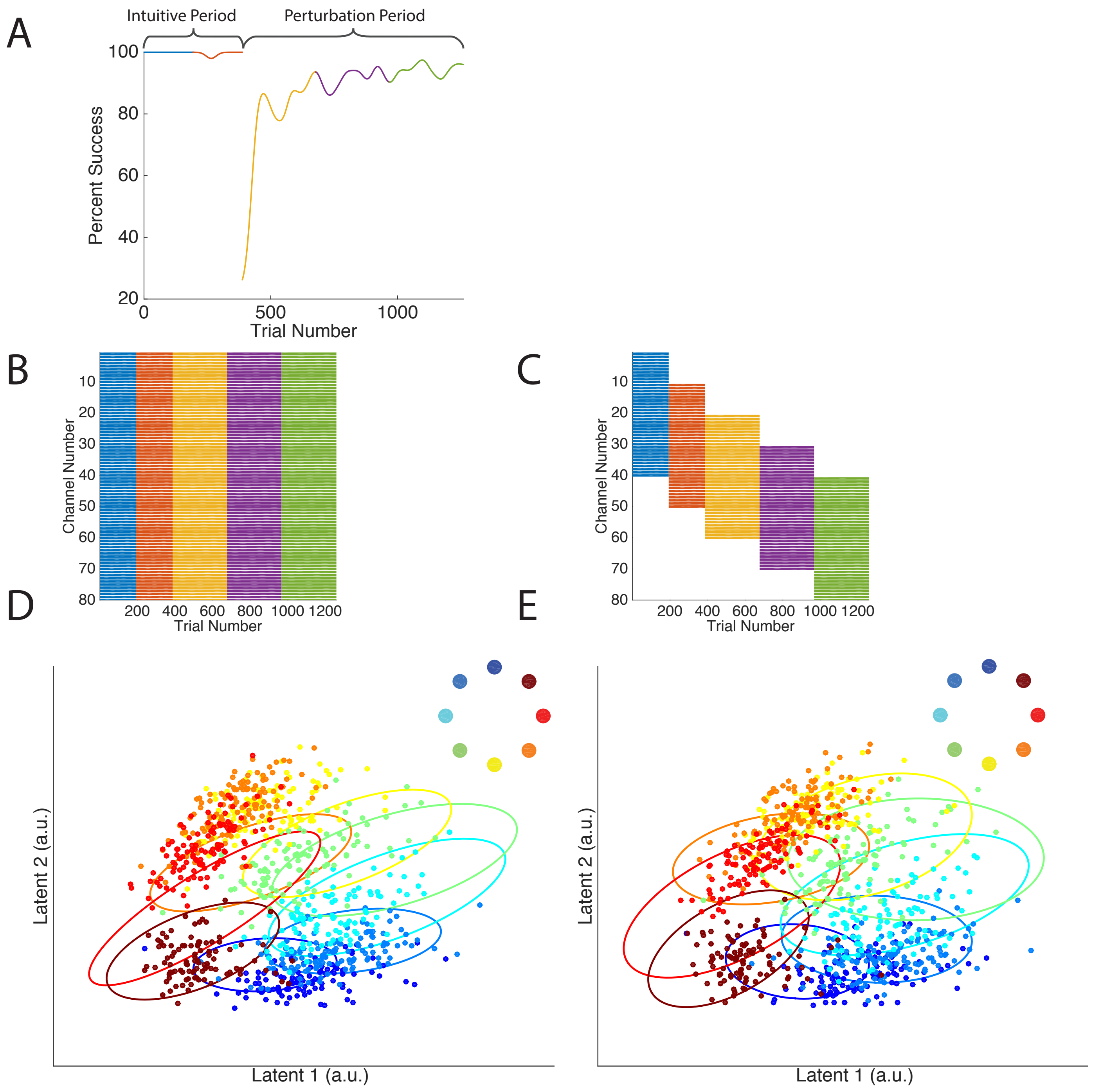


Figure 6.1: (A) Smoothed task success over time during the BCI session analyzed here. Task segments are color coded by simulated recording block. (B) and (C) Schematic depiction of electrodes which are recorded from on each trial for the simultaneous (B) and stitching (C) scenarios. Shading indicates an electrode is active for a given trial. Colors again correspond to blocks of the simulated experiments. (D) and (E). Estimated latents under the simultaneous (D) and stitching (E) scenarios. Individual points indicate latents for each trial during the perturbation period, color coded according to the target for a trial (targets shown in insets). Regions containing 95% of the probability density for inferred latents during the intuitive session are shown with ellipses. The coordinate system for latents in (E) has been transformed to allow for the best comparison with panel (D).

We now present a set of simulations to quantify the performance of stitching. We imagine a scenario in which we are able to record from a nominal number of neural units each day, but due to instability, the membership of this set changes from day-to-day. If we desire to fit an FA model to this data and infer underlying low-dimensional state for trials on each day we have two options. First, if neural units come and go slowly, there may be a sufficiently large set which are present across all recordings. In this case, we can discard any unit which was not present on all days, and fit an FA model using only those units that were common across all days. Of course, if we record for long enough or if neural units turn over at a fast enough rate, there will be no common units. The second option is to use stitching to fit a joint FA model to every unit that we record from across all days. Using simulated data, we now compare these two approaches.

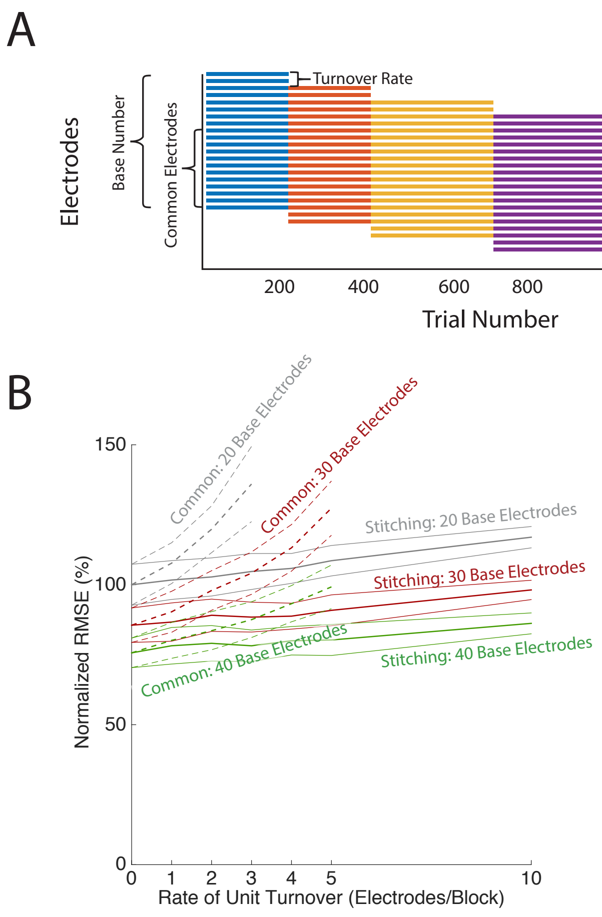


Figure 6.2: Error of estimated latents under common and stitching scenarios. (A) Scenarios were parameterized by the base number of electrodes recorded from on any trial and the turnover rate of electrodes between blocks, resulting in a given number of common electrodes between all blocks. Note for large turnover rates, there were no electrodes common between all blocks. (B) Normalized error for differing base values with increasing turnover rates. Dashed lines are for scenarios using common electrodes; solid lines for stitching scenarios. Thick lines represent means, with thin lines representing standard error for the mean for 100 random simulations of each scenario.

We generate data for our simulations according to a statistical model fit to the data described in section 6.1.1. This is a single-day learning experiment, and therefore we simulate electrodes as coming and going in blocks of trials in single simulated days. We break up each simulation into five blocks (two in the intuitive period and three in the perturbation period), and simulate multiple stitching scenarios as described in section 6.1.1. Fig. 6.2A provides an illustration of how we simulate the data. We simulate scenarios where the number of electrodes that we can record from on any trial changes. We refer to this as the base number of electrodes and vary it between 20, 30 and 40. For each value of base electrodes we simulate experiments where there is no electrode turn over between blocks.

This represents an ideal scenario where channels were completely stable. In reality, experimentalists will have to contend with unstable recordings. Therefore, we also simulate scenarios where electrodes come and go at different rates. We simulate a turn-over rate of 1,2,3,4,5 and 10 electrodes per block for each base value. For each base value and turn over rate, we perform 100 random simulations as described in section 6.1.1. For each simulation, we fit FA models using stitching and infer the underlying latents. When the number of common electrodes is greater than the dimensionality of the FA model we seek to fit, we also fit FA models and infer latents with the common units. After inferring latents for a simulation, we measure error of the estimated latents as described above.

The simulated results are presented in Fig. 6.2B. Error reported in the figure has been normalized by dividing error values for all scenarios by the mean error achieved by the simulated stitching experiments for a base value of 20 electrodes with no turn over. Plots for the common scenarios are shorter than those for the stitching scenarios because there were no common units for a turn over rate of 10 electrodes per block. Three trends are apparent in the figure. First, in ideal scenarios when there is no turn over, recording from more electrodes at a time improves estimation of low-dimensional state. This is expected. Second, estimation error in the common scenarios increases with turn over rate. This highlights the limitation of using only those electrodes which are common across all recordings: by reducing the number of electrodes available to infer latent state on any trial, estimation error increases. Third, by using stitching, we are able to estimate latents with accuracy near that of the ideal scenarios where there is no electrode turnover. This is true, even for scenarios where turn over is fast enough that there are no common electrodes across blocks. The slight upward trend in error with turnover rate for the stitching scenario is likely due to the increase in the number of parameters in the FA model as more electrodes are observed.

6.2 Stitching to Study Multi-Area Communication

Studies relating neural activity among different brain areas have been carried out with techniques such as MEG and fMRI (e.g., [111–113]), EEG (e.g., [114, 115]) and traditional electrophysiological techniques ([116–119]). In this section, we focus on studying interactions of neurons in two areas at the population level. We seek to model interactions between not individual neurons but patterns of activity between two neural populations.

Recently Semedo et al. [96] proposed extensions to canonical correlation analysis (CCA) to model interactions between latent brain state, as opposed to individual neurons, in two brain areas. This reduces the number of parameters required to model interactions among areas, enhancing interpretability and reducing the proneness of models to overfitting. In addition to this technical advance, work by Kaufman et al. has found that in motor cortex neural activity during movement occupies a subspace that is largely orthogonal to the subspace for planning neural activity [10]. This suggests that at least in some brain areas, communication may take place in particular neural subspaces.

In this section, we present two methods for identifying subspaces in which neural data in one population covaries with or is predictive of neural activity in a second subspace. While these methods could be applied to any two neural populations, we are particularly interested in identifying subspaces in two different brain areas. We propose that the identification of these subspaces could enable exciting new experiments. For example, using optogenetics in online experiments, behavioral effects may manifest themselves if neural activity could be disrupted when it is detected to lie within such subspaces.

Important for our proposed use of stitching, in such online experiments, after the subspaces are identified, it may be sufficient to monitor the activity of neural population of only one area. For example, if we wish to disrupt the transfer of information from primary visual cortex (V1) to secondary visual cortex (V2), identifying when V1 activity has entered the subspace for communication with V2, would still allow us to potentially selectively disrupt information flow between these two areas.

Critical to the success of such experiments is the identification of subspaces themselves. This would seem to require the ability to simultaneously record from many neurons in both areas, since we seek subspaces which carry information between areas at the neural population level. In the work that follows, we will explain how stitching experiments could be carried out to allow for the identification of these subspaces when neural activity from the entirety of both populations of interest cannot be recorded simultaneously. We will then present simulated stitching experiments, based on data that was simultaneously recorded from populations of neurons in V1 and V2, to examine the effectiveness of such an approach.

6.2.1 Methods

Notation

We will refer to a vector of spike counts recorded in area 1 with the notation $x_t^1 \in \mathbb{N}^{M_1}$, where M_1 is the number of units in area 1. Similarly, $x_t^2 \in \mathbb{N}^{M_2}$ will be a spike count vector for M_2 units in area 2. We assume x_t^1 and x_t^2 are recorded simultaneously when they are indexed by the same value of t. We will use the matrices $X_1 \in \mathbb{N}^{T \times M_1}$ and $X_2 \in \mathbb{N}^{T \times M_2}$ to collect a set of T observation vectors from both areas. We will let $\Sigma_{11} \in \mathbb{R}^{M_1 \times M_1}$ be the covariance matrix for the counts in X_1, and $\Sigma_{22} \in \mathbb{R}^{M_2 \times M_2}$ be the covariance matrix for the counts in X_2. Similarly, $\Sigma_{12} \in \mathbb{R}^{M_1 \times M_2}$ will capture the covariance between counts in X_1 and X_2. In all the work below, we will assume X_1 and X_2 have been mean centered.

Methods for Identifying Covarying Subspaces

We now present two methods for identifying subspaces in which neural activity in one neural population covaries with or is predictive of neural activity in a second. We will introduce each in the context of analyzing recordings of simultaneous neurons. We will explain what each is optimized to find and how this may be useful for analyzing neural recordings. After presenting the basics of each method, we will describe how each can be used in a stitching scenario.

In all of the stitching scenarios we consider, we assume that we record simultaneously from a population of neurons in one area while recording from a neural population in a second area in blocks. Using the scenario we simulate below as an example, a multielectrode array could be used to constantly record from a set of units in V1, while linear probes are moved from time to time to record from different populations of neurons in V2. In this way, every set of V2 neurons recorded would be simultaneously recorded with the same set of V1 neurons, but the blocks of V2 neurons themselves may or may not have units in common between them.

Reduced Rank Regression Reduced Rank Regression (RRR) [120] seeks to find a J dimensional subspace in area 1 which best predicts neural activity in area 2. Equivalently, RRR can be thought of as finding a subspace of neural activity in area 2 of dimensionality J which can be best predicted by neural activity in area 1. Under RRR, we seek a matrix A such that

$$||X_2 - X_1 A||_F^2 \tag{6.2}$$

is minimized subject to the constraint that $\textbf{rank}\{A\} \leq J$. In our work, we derive projection vectors $d_1^1, \ldots, d_J^1$ in the space spanned by neural activity for area 1, and corresponding vectors $d_1^2, \ldots, d_J^2$ in the space spanned by the neural activity for area 2 such that $d_1^1, \ldots, d_J^1$ and $d_1^2, \ldots, d_J^2$ are the first left and right singular vectors of A. The projection vectors $d_1^1, \ldots, d_J^1$ then form an orthonormal basis for the subspace of neural activity in area 1 that is used to predict neural activity in area 2. Similarly, $d_1^2, \ldots, d_J^2$ form an orthonormal basis for the predictions of area 2 neural activity.

RRR may be useful for studying communication if we think a signature of communication between two brain areas is the ability to predict spike counts in one area based on spike counts in another. RRR discounts subspaces which may yield highly correlated neural activity in two areas if activity in the second subspace is poorly predicted from activity in the first. This may be important, for example, when looking for attentional affects, where the attentional drive from one area may only weakly bias the firing rate of any particular neuron in a second. Under this scenario, there may not be a pair of subspaces capturing highly correlated activity between the two areas, but by looking for subspaces which are good for prediction, RRR may still identify subspaces which capture the attentional effect between the two areas.

The matrix A in RRR can be calculated in a two-step manner [120]. In the first step, A_{Full} is calculated using the standard least squares equations as $A_{\text{Full}} = \Sigma_{11}^{-1}\Sigma_{12}$. In the second step, the eigendecomposition of a matrix $H = \Sigma_{12}^T \Sigma_{11}^{-1} \Sigma_{12}$ is found. If E contains the top J eigenvectors of H, A is then calculated as

$$A = EE^T A_{\text{Full}}. \tag{6.3}$$

In the stitching scenarios that we consider, Σ_{11} and Σ_{12} can be estimated entry-wise. This is possible because all neurons in area 1 are recorded simultaneously in each recording block, providing a set of data with which to estimate Σ_{11}. Similarly, each block records from some neurons in area 2 while recording from all neurons in area 1. This provides data to estimate the appropriate entries in Σ_{12}. Therefore, the more sophisticated algorithm presented in chapter 4 to combine recordings is not required. Nonetheless, we include this scenario under the larger umbrella of "stitching" as we still seek to perform analyses that would be not be possible without combining recordings from overlapping neural populations.

Canonical Correlation Analysis Canonical correlation analysis (CCA) seeks to find a set of projection vectors $d_1^1, \ldots, d_J^1$ in the space spanned by neural activity for area 1, and corresponding projection vectors $d_1^2, \ldots, d_J^2$ in the space spanned by the neural activity for area 2, such that

$$\text{corr}\left(X_1 d_j^1, X_2 d_j^2\right)$$

is maximized subject to $(X_1 d_j^1)^T X_2 d_i^2 = (X_1 d_j^1)^T X_1 d_i^1 = (X_2 d_j^2)^T X_2 d_i^2 = 0$ for all $i \neq j$ [121]. That is CCA seeks to find a set of pairs of projection vectors such (1) the projected neural activity of each pair is maximally correlated within pairs and uncorrelated across pairs and (2) the projected data within a single area is uncorrelated. By convention we will order the pairs of directions so that corr $\left(X_1 d_1^1, X_2 d_1^2\right) \geq \ldots \geq$ corr $\left(X_1 d_J^1, X_2 d_J^2\right)$.

In contrast to RRR, CCA discounts the amount of variance in total neural activity contained in the two subspaces. This would be important if we believe a tight correlation is a signature of neural activity even if the subspaces with highly correlated neural activity contain little of the overall neural variability. A scenario in which CCA may be preferable over RRR would arise, for example, if communication between brain areas took place between a select number of highly correlated neurons in each population.

Let $F = \Sigma_{11}^{-1/2} \Sigma_{12} \Sigma_{22}^{-1/2}$. The projection vectors $d_1^1, \ldots, d_J^1$ and $d_1^2, \ldots, d_J^2$ can be computed in a two step manner. In the first step, if $U \Lambda V^T$ is the singular value decomposition of F, the left and right singular vectors contained in the columns of U and V are found. We assume that the columns of U and V are ordered from left to right to contain the singular vectors corresponding to the largest to smallest singular values. In the second step, projection vectors are calculated as

$$d_j^1 = \Sigma_{11}^{-1/2} U(:, j)$$
$$d_j^2 = \Sigma_{22}^{-1/2} V(:, j).$$

Unlike RRR, the calculations to find projection vectors for CCA requires the full covariance between all pairwise units recorded in both areas. Thus, there will be no stitching scenarios where we are fortunate enough to be able to simply compute the entries of the portion of the covariance matrix that we need in a trivial way. For the stitching scenarios we consider here, we will be able to compute Σ_{11} and Σ_{12} entry-wise from the recorded data, as with RRR, but we will not be able to estimate Σ_{22} in this way. However, if there is overlap between the blocks of units recorded in area 2 we can use the algorithm in chapter 4 to fit an FA model to this data. Using the notation from that chapter, let $\hat{C}$ and $\hat{\Psi}$ be the loading matrix and matrix of private noise returned by fitting an FA model to V2 data in this way. We can then estimate $\hat{\Sigma}_{22} = \hat{C}\hat{C} + \hat{\Psi}$. We can use $\hat{\Sigma}_{22}$ calculated in this way, as well as the estimates for Σ_{11} and Σ_{12} calculated entry-wise in the equations above to find the projection directions for CCA.

Neural Recordings

In this work, we will analyze neural activity simultaneously recorded from 165 single and multi-units in V1 and 34 single and multi-units in V2. Although recordings were made simultaneously, we will generate simulated stitching scenarios by assuming we can only record from a small number of neurons in V2 at once. We analyze recordings performed by Adam Kohn's lab at the Albert Einstein College of Medicine from an experiment in which an anesthetized rhesus macaque was shown gratings in one of eight orientations. Each grating was displayed for 1280 milliseconds, followed by an intertrial interval of 1500 milliseconds. Each grating was shown 400 times, for a total of 3200 trials. Details of the recordings can be found in [122]. The 165 and 34 units in V1 and V2 constitute all recorded units that were not marked as noise or junk in the original recordings.

In all of the work that follows, we analyze only neural data during the stimulus presentation, discarding neural activity in the intertrial intervals. For each trial, we form spike count vectors by counting spikes in non-overlapping bins during the entire duration of stimulus presentation. In the results below, we specify the bin sizes used. Before performing any of the analyses below, we subtract average responses to each grating from the spike count vectors to produce vectors of residual spikes counts, as we seek to identify covariability between V1 and V2 that is non-stimulus driven. This is somewhat unrealistic for the stitching scenarios we simulate, as this allows us to calculate average stimulus responses for all units using using all trials of an experiment. In practice, for a stitching experiment, the average response for a neural unit will have to be estimated using only the trials it was recorded on, which when stitching will be less than the total number of trials in a the full experiment. However for analyses with a large number of trials, the average responses that are calculated in stitching scenario should differ little from those we calculate with this procedure, and this simplifies the presentation of the methods and results below.

Evaluation with Simulated Stitching Scenarios

Simulations We simulate stitching scenarios using the real data described above. Specifically, although all recorded units were in reality recorded simultaneously, we use the recorded spike counts on each trial and simulate scenarios where we assume we have the ability to record simultaneously from all V1 units in the original recordings but only B units at once in V2. We assume that in each block of an experiment, we can move electrodes in V2 so that we continue to record from some neural units from the previous block but also record from some units we haven't sampled before. See Fig. 6.3A and Fig. 6.5B for illustrations of this general sampling scheme. Each scenario is then characterized by T, the total number of trials in a simulated experiment, B the number units that we record at once in V2, ${M_2}'$ the total number of units sampled in V2 over the course of the experiment, and O the number of common V2 units that we record from in two sequential blocks. The prime on M_2 indicates that a simulated experiment may sample less than the full 34 V2 units in the original recording. Given these parameters, we randomly generate a stitching scenario from the prerecorded data in the following manner.

1. The original 3200 trials of data are down sampled to T trials. Trials are down sampled in a deterministic fashion respecting the balance of grating orientations and original ordering of trials. Trial ordering is respected to preserve slow non-stationarities which may be present in the data.
2. Given ${M_2}'$, O and B, the number of blocks we simulate recording from sets of V2 neurons in, K, is calculated as $K = ({M_2}' - O)/(B - O)$. We always select ${M_2}'$, O and B so that K evaluates to a whole number.
3. Given the down sampled trials from step (1) and K, trials are assigned to each block, again respecting the original ordering of trials. If T is not divisible by K, one extra trial is assigned as needed to recording blocks occurring earlier in the simulated experiment.
4. Sets of units from $V2$ to record from in blocks are randomly generated as parameterized by ${M_2}'$, O B and K.
5. Finally, simulated stitching data is generated by retaining only spike counts from the units simulated as being recorded from in each block of the down-selected trials.

Evaluation We now describe how we evaluate the performance of CCA and RRR in our simulated stitching scenarios. For each simulated stitching dataset, the set of simulated trials is broken up into 10 folds. Folds are generated so that trials from each simulated recording block are represented as evenly as possible in each fold. The parameters for CCA or RRR are then learned from data on 9 of the 10 folds. Finally, subspaces identified by CCA and RRR are evaluated on the held out fold of data as described below. For each simulated dataset, we do this 10 times, using each separate fold of data as the test set.

Each time CCA or RRR is fit, they return subspaces for both V1 and V2. The evaluation metrics we use focus on the subspace for V1. There are two reasons for this. First, in all of the simulated experiments we perform, we record from all V1 neurons. Thus, subspaces for V1 neurons across stitching scenarios can be meaningfully compared. In contrast, it is not clear how to meaningfully compare subspaces for V2 across different stitching scenarios which may record from different numbers of V2 neurons. Second, in keeping with the motivation presented in the introduction, we envision experiments where after the subspaces are identified with stitching, it is sufficient to monitor the activity in only one area. In these settings, the quality of the V1 subspace can be judged in a functional manner: two subspaces in V1 are equally good if they capture the same amount of correlation with V2 (CCA) or predict activity in V2 with the same accuracy (RRR).

Let $X_{\text{test, 1}}$ be spike counts from all units in V1 in a held out fold of data. Let $X_{\text{test, 2}}$ be spike counts from all 34 units that were recorded in V2 for the same fold of data. Note that no matter how many units in V2 a simulated stitching scenario recorded from, $X_{\text{test, 2}}$, always contains counts from all V2 units in the prerecorded data.

Let $P_{\text{fit}} \in \mathbb{R}^{M_1 \times M_1}$ be a projection matrix which projects activity in V1 into the subspace identified by RRR. We then evaluate the quality of the subspace parameterized by P_{fit} for RRR by solving the following optimization problem:

$$E_{\text{min}} = \min_{A \in \mathbb{R}^{M_1 \times M_2}} ||X_{\text{test, 2}} - X_{\text{test, 1}} P_{\text{fit}} A||_F^2, \tag{6.4}$$

subject to the constraint that **rank**$\{A\} \leq J$, where J is the dimensionality of the subspace spanned the columns of P_{fit}. In words, E_{min} gives the best prediction error that we can achieve on the test data in V2 if we project our test data in V1 to the subspace identified by RRR fit on the training data. In this way we are asking how good the subspace for V1 returned by RRR is at predicting the activity of all units in V2. In the results we present below, we normalize E_{min} by $||X_{\text{test, }2}||_F^2$ and subtract the result from 1 to arrive at percent variance explained.

To evaluate a CCA model that returns J pairs of univariate subspaces, we let P_{fit} be a projection matrix to the subspace spanned by $d_1^1, \ldots, d_J^1$ fit on the training data. We the solve the following problem. For $j \in 1 \ldots J$, we maximize

$$\text{corr}\left(X_{\text{test, }1} P_{\text{fit}} d_j^{1\prime}, X_{\text{test, }2} d_j^{2\prime}\right) \tag{6.5}$$

subject to $(X_{\text{test, }1} d_j^{1\prime})^T X_{\text{test, }2} d_i^{2\prime} = (X_{\text{test, }1} d_j^{1\prime})^T X_{\text{test, }1} d_i^{1\prime} = (X_{\text{test, }2} d_j^{2\prime})^T X_{\text{test, }2} d_i^{2\prime} = 0$ for all $i \neq j$. The prime on $d_1^{j\prime}$ and $d_2^{j\prime}$ indicates they will in general differ from the those learned on the training data. In other words, we solve the original CCA problem on the test data when $X_{\text{test, }1}$ is constrained to the subspace identified by fitting CCA on the training data. Given a set of $d_1^{1\prime}, \ldots, d_J^{1\prime}$ and $d_1^{2\prime}, \ldots, d_J^{2\prime}$, we evaluate the quality of the original J dimensional subspace by measuring

$$\text{corr}_{\text{max},J} = \text{corr}\left(X_{\text{test, }1} P_{\text{fit}} d_J^{1\prime}, X_{\text{test, }2} d_J^{2\prime}\right).$$

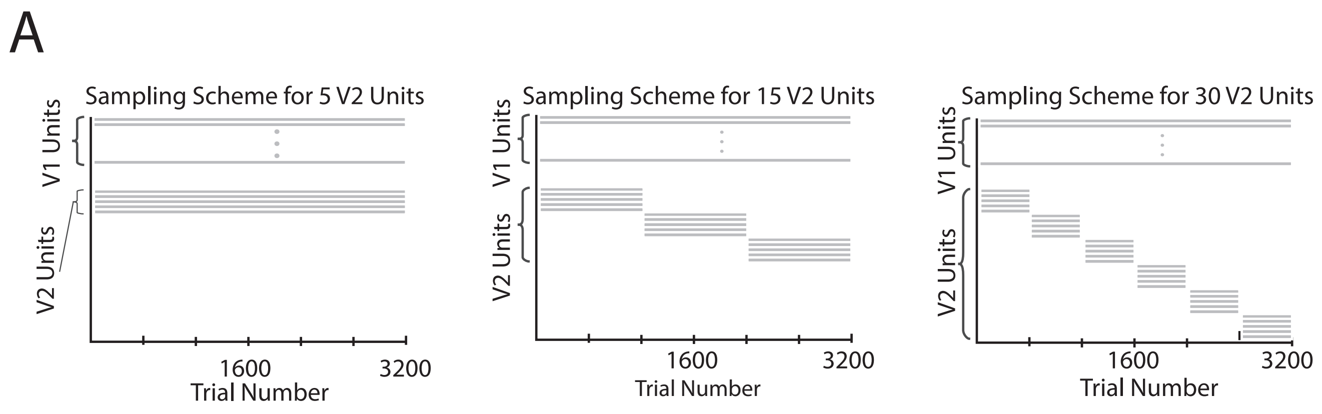
That is we measure the quality of a J dimensional subspace by the smallest correlation captured in the pair of subspaces fit to the test data when the test data in area 1 is constrained to the subspace identified by fitting CCA on the training data. We chose the smallest correlation as it lower bounds the value of correlations for all projection pairs found between two subspaces. Similar to the evaluation metric for RRR, this allows us to ask how good the V1 subspace fit to training data is at predicting correlations among the full set of V2 neurons in the test set of data.

6.2.2 Results

We simulate stitching scenarios from the prerecorded data. In each scenario we simulate, we imagine that we can record simultaneously from all 165 V1 units in the original recordings but can only record from a limited number of V2 units simultaneously. Continuing with the motivation presented in the introduction, we would like to identify a subspace in V1 neural activity which is potentially important for communication with V2. To do so, we must record from at least some neural units in both areas simultaneously. Faced with the ability to record only a limited number of V2 units, we must then decide if we record from the same limited number of V2 units over the entire duration of an experiment or if we move our recording devices and use stitching to sample more V2 units throughout the experiment. As we sample more units, a more holistic picture of how populations of neural units in V2 covary with neural units in V1 should emerge. However, at the same time, we also increases the number of parameters in the model without increasing the amount of data available for model fitting.

Fig. 6.3 presents results for RRR where we simulate recording from $B = 5$ neural units at once in V2 in non-overlapping blocks ($O = 0$). For this analysis, we bin neural activity in each trial in 20 millisecond bins. Fig. 6.3A illustrates the three sampling scenarios we consider. In the base scenario, we simulate recording from the same $M_2' = 5$ units for all 3200 trials (left illustration of Fig. 6.3A). We also simulate recording from $M_2' = 15$ (3 blocks of 5 units each, middle illustration of Fig. 6.3A) and $M_2' = 30$ (6 blocks of 5 units each, right illustration of Fig. 6.3A) units in V2. We simulated 100 repetitions of each of these scenarios as described in section 6.2.1. For each repitition, we fit and evaluate RRR models with dimensions ranging from 1 to the number of V2 neurons sampled. We evaluate the fit of these models as described in section 6.2.1. For each simulated dataset, this returns the amount of variance explained in each of the 10 folds of test data for RRR models of each dimension. We take the average across all folds and use this to quantify the average percent variance explained by a RRR model of each dimension on a given repetition of data. Fig. 6.3B then plots the mean of these averages over all 100 simulated repetitions for the three scenarios.

As can be seen in Fig. 6.3B, RRR fits obtained with stitching which sample more units return estimates of subspaces in V1 that allow for better prediction of activity in the full V2 population, showing the utility of stitching. This did not have to be the case. It is possible that as a recording device is moved more frequently during an experiment to sample more neurons, the number of parameters in the model would grow such that overfitting would become a problem. To illustrate this general concept, we run an additional simulation below with a reduced number of simulated trials in each scenario.


A
Sampling Scheme for 5 V2 Units
Sampling Scheme for 15 V2 Units
Sampling Scheme for 30 V2 Units
V1 Units
V2 Units
1600
3200
Trial Number

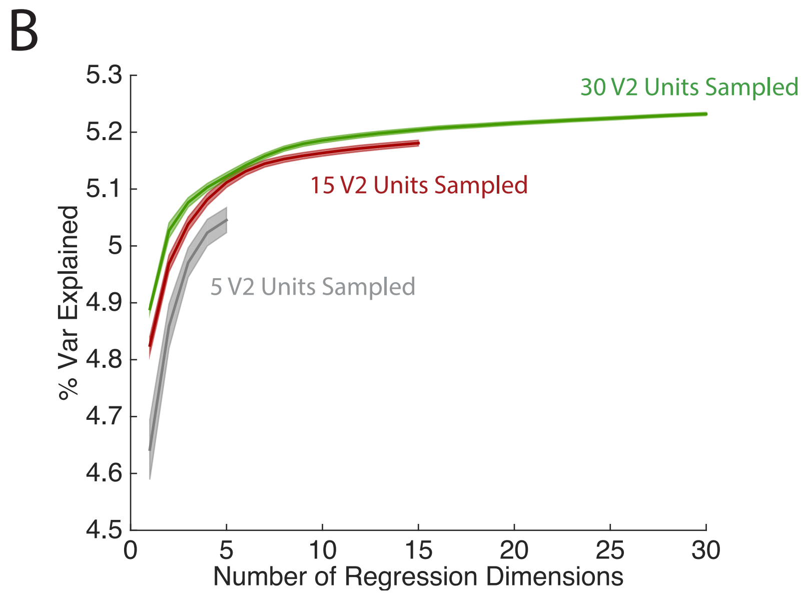


Figure 6.3: (A) The three stitching scenarios simulated in simulations. In each scenario we simulate a trial budget of 3200 trials. All V1 units in the prerecorded data were simulated as being recorded from in each trial. Randomly selected sets of 5 V2 units were simulated as being recorded in blocks. The number of blocks, and therefore total number of V2 units sampled, varied by scenario. (B) Average percent variance explained of V2 neural activity obtained from RRR models fit to test data when neural activity in V1 was constrained to lie in the V1 subspace returned by RRR models fit to training data. See text for details. Results are plotted for RRR models of varying dimension for each scenario. Solid lines are mean performance across all simulated repetitions. Shading represents 95% confidence intervals.

We also add a note for interpretive caution. The monotonic increase in percent variance explained for each scenario is a direct consequence of searching in larger subspaces in the test set of V1 neural data to explain the test set of V2 neural activity. It is likely, therefore, that increases in percent variance explained to the right of each plot are statistically insignificant. The statistical significance of these increases could be assessed via permutation testing, where the pairing of neural recordings in V1 and V2 is randomized. This would allow us to truncate the plots in panel B for dimensions which do not explain statistically significant increases in percent variance explained. However, for computational reasons we do not present those results here.

Fig. 6.3 illustrates that for RRR models of fixed dimension, given a sufficiently large trial budget, increasing the number of neurons sampled through stitching allows RRR to find V1 subspaces better suited to predicting V2 neural activity. However, for any fixed trial budget, sampling too many neurons will increase the number of parameters in a model such that overfitting becomes a problem. To illustrate the tradeoff between the number of neurons sampled and the parameters that must be fit, we rerun the same analysis but down sample to a total of 208 trials, which is 26 per grating orientation (Fig. 6.4). As we move from sampling 5 to 15 V2 units, RRR returns V1 subspaces containing neural activity that is better able to predict V2 activity. This is a result of sampling more V2 units. However, as the number of units sampled is increased to 30, there is no longer sufficient training data to prevent overfitting for these models with more parameters and the quality of the fit decreases. Given a set of stitching data, overfitting can be combatted by reducing the number of dimensions the RRR model returns. In a real stitching scenario, we propose that leave one out neuron prediction error [9] can be used to assess the proper dimensionality of a RRR model given set of stitching data.

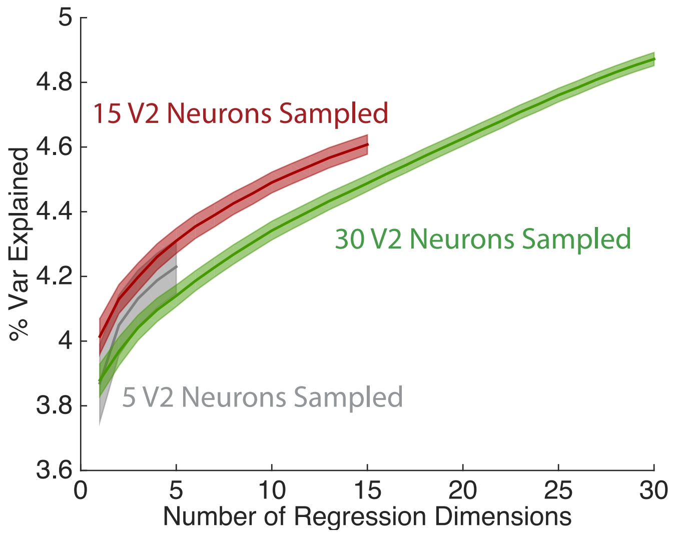


Figure 6.4: Evaluation of RRR under the same stitching scenarios as presented in Fig. 6.3, when scenarios are simulated with a trial budget of 208 trials. Results are plotted in the same manner as those in Fig. 6.3B.

We now present results for CCA. To increase the magnitude of correlations between neurons, we count spikes in 500 millisecond bins in each trial. Just as we did for RRR, we simulate three stitching scenarios as described in section 6.2.1. In these scenarios we simulate recording from $B = 5$ units in V2 simultaneously, but now allow for overlap of $O = 4$ units between blocks. This overlap allows us to apply the method described in section 6.2.1 to estimate the covariance matrix for the sampled neurons in V2 before calculating pairs of CCA projection vectors. As shown in Fig. 6.5A, we then simulate recording from $M_2' = 5$ (1 block) units consistently throughout an experiment and moving our recording device to record from $M_2' = 10$ (6 blocks) and $M_2' = 15$ (11 blocks) units in V2 in total in this way. With the exception of differences in these parameters, simulations were performed just as they were above for RRR. For computational reasons, for each scenario, we only perform 32 random repetitions.

For each simulated repetition, we fit and evaluate CCA models to the simulated data. For the baseline scenario when we simulate recording from the same 5 neural units for the entire experiment, we can fit CCA models without the need for our stitching algorithm (as there is no missing data). For the scenarios when we sample 15 and 30 units in V2, we use stitching with FA models with 3 latents to estimate the covariance matrix for the V2 neurons and then apply CCA as described in section 6.2.1. Evaluation is performed as described in section 6.2.1. For each simulated dataset, this returns $\text{corr}_{\text{max},J}$ values for each of the 10 folds of data for CCA models of each dimension. As we did with RRR, we take the average across all folds and use this to quantify the average $\mathrm{corr}_{\mathrm{max},J}$ value for a CCA model of each dimension on a given repetition of data. Fig. 6.5 then plots the mean of these averages over all 32 simulated repetitions for the three scenarios.

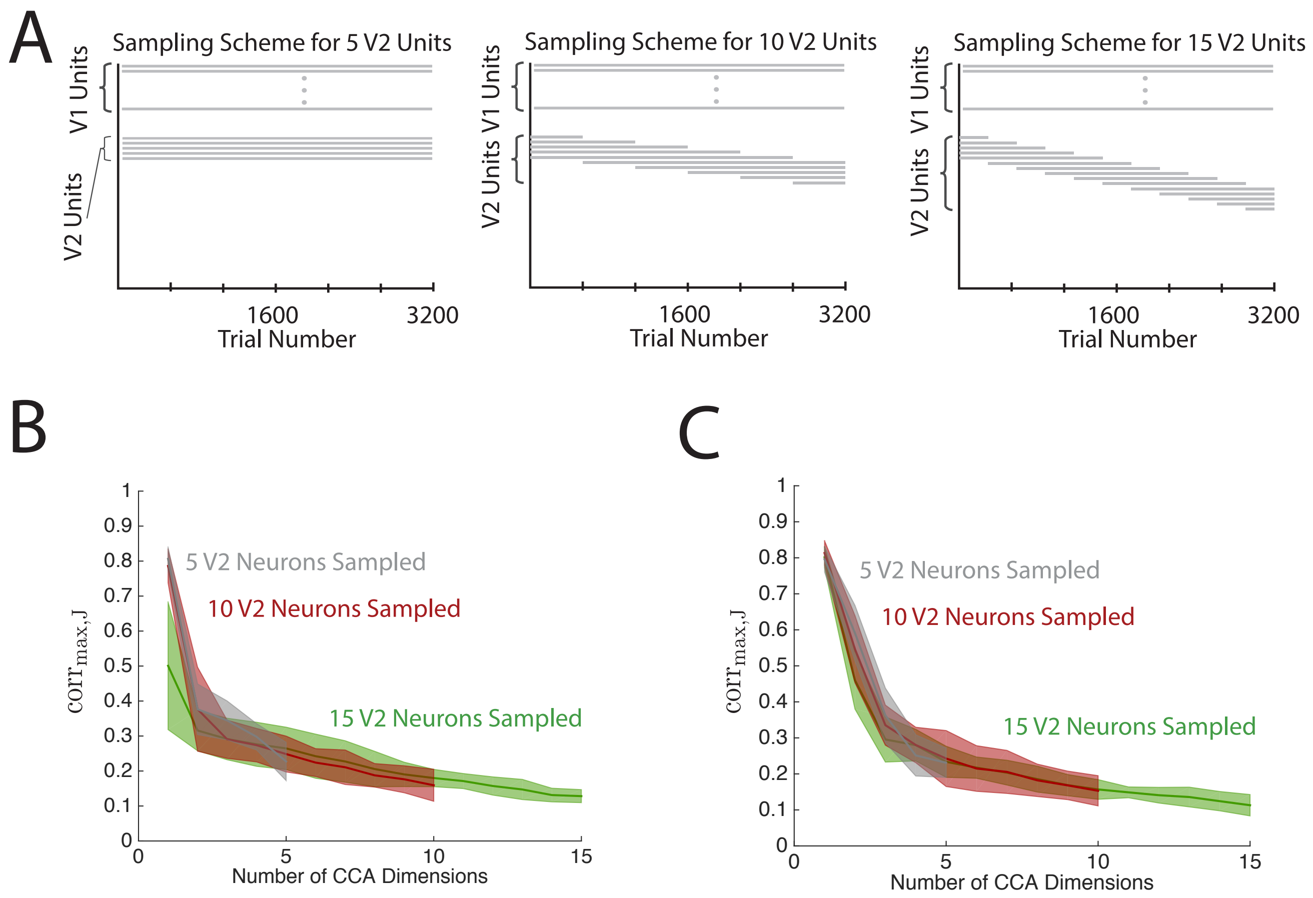


Figure 6.5: (A) The three stitching scenarios simulated in simulations, plotted as in panel A of Fig. 6.3. A trial budget of 3200 trials was again used. All V1 units were recorded on each trial. V2 units were recorded in overlapping blocks of 5 units. (B) Average value of $\mathrm{corr}_{\mathrm{max},J}$ for subspaces returned by CCA models of varying dimension for each scenario. Solid lines are mean performance across 32 simulated repetitions. Shading represents 95% confidence intervals. (C) Average value of $\mathrm{corr}_{\mathrm{max},J}$ for subspaces returned by CCA models of varying dimension for each scenario when the covariance matrix for V2 neurons is assumed to be diagonal. Results are reported for 100 random repetitions of each scenario.

Fig. 6.5B presents the results of this analysis and indicates that CCA models fit under the baseline scenario and stitching scenarios with 10 sampled units in V2 perform very similarly. However, CCA models fit under stitching scenarios with 15 sampled units, particularly when CCA is used to return a one dimensional subspace, perform less satisfactorily. There are three probable reasons for this. First, a CCA model for 15 V2 units has more parameters than CCA models for 5 or 10 V2 units and overfitting could affect the results. Second, the stitching algorithm to learn the parameters of the FA model must align estimates of the loading matrices for each block. This process can become fragile as the number of blocks used to stitch increases. Third, in this particular dataset the covariance among neurons in V2 seems to minimally affect the results of CCA.

We find evidence that the covariance structure among the recorded V2 neurons is weak enough to minimally affect CCA results in two ways. First, we rerun the same simulations but instead of running our stitching algorithm, we assume Σ_{22}, the covariance matrix for V2, is diagonal. This allows us to estimate the variance of each neural unit entry-wise in the stitching scenarios. Because we no longer need have the computational burden of fitting FA models, we increase the number of random repetitions for each scenario to 100. We quantify the fit of the CCA models exactly as in the previous analysis and plot the results in Fig. 6.5C. Using Σ_{22} estimated in this way, we find that performance of CCA when sampling 15 units in V2 is recovered. This is consistent with the hypothesis that there is little covariance structure among V2 neurons and the stitching algorithm in simulations sampling 15 V2 neurons becomes fragile.

Second, to further assess the importance of the covariance structure between V2 neurons we run a final set of simulations. In each simulation, we sample all neurons of the V1 and V2 units in the original recordings in each trial. As with previous results, we perform 100 random repetitions, where because we sample all neurons in each repitition the only difference in repetitions is which trials are assigned to which folds for the purposes of cross-validation. We then fit CCA models in these simulations in two ways. In the first way, we fit CCA models using the full sample covariance matrix among V2 neurons. In the second, we assume the covariance matrix among V2 neurons is diagonal. Performance is again quantified in the same way as the previous two analyses. As shown in Fig. 6.6, the results when CCA is fit in both ways are nearly identical.

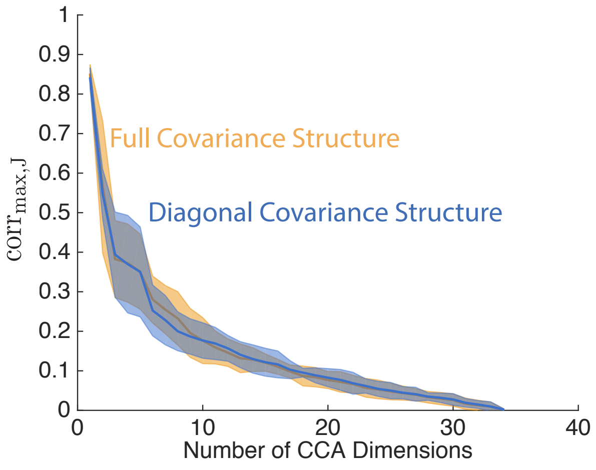


Figure 6.6: Average value of $\mathrm{corr}_{\mathrm{max},J}$ for subspaces returned by CCA models of varying dimension under the assumption that the covariance matrix for V2 was full and diagonal for scenarios simulating recording from all V1 and V2 units in the original recordings simultaneously. Results are reported for 100 random repetitions of each scenario. Solid lines are average performance across 100 simulated repetitions. Shading represents 95% confidence intervals.

Chapter 7

Deterministic Symmetric Positive Semidefinite Matrix Completion via Nuclear Norm Minimization

For the last chapter of this thesis, we turn once again to theory. In chapter 3 we established that a symmetric positive semidefinite (SPSD) matrix can be recovered when a set of its principal submatrices meeting certain sufficient conditions is observed. In contrast to much of the previous matrix completion literature, these sufficient conditions do not require the underlying matrix to be incoherent. Intuitively, incoherence rules out matrices which can hide most of their mass in a small subset of unobserved entries.

Intrigued by this observation, we ask if we can prove that a common technique for matrix completion, nuclear norm minimization, will also recover SPSD matrices under the sufficient conditions presented in chapter 3. In this chapter we will show that this is in fact the case. This is important for two reasons. First, it establishes that nuclear norm minimization can be applied to the completion of an important class of coherent matrices with structured missingness. Second, we provide empirical evidence that nuclear norm minimization may provide a more robust method of completing SPSD matrices in the presence of noise. Both of these findings have important practical implications.

7.1 Background

The problem of recovering a low-rank matrix through solving a nuclear norm penalized objection has been extensively studied (e.g., [26, 28–30, 86, 87]). In the noiseless setting, the problem is formulated as finding the matrix of minimum nuclear norm which agrees with all of the observed entries of the matrix we desire to recover [28]. In the noisy setting, the problem has been refered to as the matrix lasso [29, 30] and is formulated as optimizing an objective which trades off reconstruction error for the observed entries of a matrix and the nuclear norm of the returned matrix.

Incoherence is a common assumption which appears in much of the existing nuclear norm matrix completion literature (e.g., [26, 28–30]). Incoherence is a measure of how close any singular vector of a matrix is to a standard basis vector. As a motivating example, consider the problem of recovering a matrix whose entries, except for one, are all zero. As pointed out by Candes and Recht [28], it would be difficult to reconstruct this matrix without sampling almost every one of its entries.

There is a body of work which has looked at relaxing the incoherence requirements. Krishnamurthy and Singh [86] use an adaptive sampling scheme to remove the requirement for incoherence on the row-space of a matrix. Chen et al. [87] show that if entries of a matrix are sampled with respect to the local coherence properties of a matrix, arbitrary coherent matrices can be recovered and they propose a two-stage sampling scheme which does not require a priori knowledge of the coherence structure of a matrix. In a separate line of work, others [88, 123] measure the complexity of a matrix without the need for incoherence assumptions and are able to bound recovery error with respect to the complexity of the underlying matrix.

In this work, we show that the conditions we present in chapter 3 are sufficient to guarantee that a coherent SPSD matrix can be recovered exactly via nuclear norm minimization. After establishing this result, we demonstrate empirically that the matrix lasso has superior noise properties to the algorithm presented in chapter 3 in two sets of simulated matrix recovery scenarios.

7.2 Preliminaries and Notation

We continue with the notation introduced in chapter 3. We again consider the problem of recovering a rank $r > 0$ matrix $A \in \mathcal{S}_+^n$, where $\mathcal{S}_+^n$ indicates the cone of n by n symmetric, positive semidefinite (SPSD) matrices. We again let $\tilde{A}$ be a noise corrupted version of A. Let $\Omega \subseteq [n] \times [n]$ be a set of ordered pairs containing the row and column indices of observed entries of A or $\tilde{A}$. Throughout this work $|\cdot|$ will indicate the cardinality of a set, so $|\Omega|$ indicates the number of observed entries of A. Define the linear operator $\mathcal{A}_\Omega : [n] \times [n] \longrightarrow \mathbb{R}^{|\Omega|}$ so that $b = \mathcal{A}_\Omega(A)$ is a vector containing all of the entries of A indexed by pairs in Ω. We will let $\|\cdot\|_*$ indicate the nuclear norm of a matrix, which is equal to sum of the eigenvalues of any SPSD matrix, or equivalently, the trace of a SPSD matrix.

Problem 1 In the noiseless setting, we propose to estimate $\hat{A}$ by solving

$$\hat{A} = \underset{M \in \mathcal{S}_+^n}{\operatorname{argmin}} \|M\|_*$$
$$\text{subject to: } \mathcal{A}_\Omega(M) = \mathcal{A}_\Omega(A).$$

That is we seek to recover a matrix of minimum nuclear norm which is consistent with all of the observed entries of A. Problem 1 is the standard approach to nuclear norm minimization [28], where we have simply constrained the search space to the cone of SPSD matrices. The cone of SPSD matrices is a convex set and therefore the convexity of Problem 1 is preserved under this modification.

Problem 2 In the noisy setting we do not expect there to be a low rank matrix which agrees with the observed entries of $\tilde{A}$. We estimate $\hat{A}$ as

$$\hat{A} = \underset{M \in \mathcal{S}_+^n}{\operatorname{argmin}} \|M\|_* + \frac{1}{2}\left\|\mathcal{A}_\Omega\left(\tilde{A}\right) - \mathcal{A}_\Omega(M)\right\|_F^2.$$

As with the noiseless setting, this is the standard matrix Lasso [29] constrained to search over the cone of SPSD matrices.

7.3 Exact Recovery of SPSD Matrices Via the Nuclear Norm

We now state the main result of this chapter.

Theorem 12. *Let A be a rank r matrix with sampled entries indexed by Ω which meets conditions $A1$ through $A4$ in chapter 3. Let $\hat{A}$ be the solution to Problem 1. Then $\hat{A} = A$.*

In the remainder of this section, we present the proof. The proof is accomplished by establishing a set of equivalency statements which will establish that Problem 1 is equivalent to the problem solved by Algorithm 1 in chapter 3. With the exception of the proof of one intermediate result, which we leave to appendix E, the entirety of the proof is relatively short and provides intuition, so we present it here. We began by introducing a third matrix completion problem. Consider estimating a matrix $\hat{A}$ as

Problem 3

$$\begin{aligned} &\hat{A} = M \\ &\text{subject to: } \; M \in \mathcal{S}_+^n \\ &\qquad\qquad\quad \mathcal{A}_\Omega\left(M\right) = \mathcal{A}_\Omega\left(A\right). \end{aligned}$$

That is we seek to find a matrix $\hat{A}$ of any rank with entries which are equal to those we observe from A. Our first intermediate result establishes that under our sufficient conditions, Problems 2 and 3 are equivalent.

Lemma 13. *Let Ω index a set of observed entries of A such that condition $A1$ and $A2$ in chapter 3 hold. Then Problems 2 and 3 are equivalent.*

We present the proof here.

Proof. Under the constraints of Problem 3, it must be that $\mathcal{A}_\Omega\left(\hat{A}\right) = \mathcal{A}_\Omega\left(A\right)$. By assumptions $A1$ and $A2$ in chapter 3, all the diagonal elements of A are indexed in Ω, so we have

$$\textbf{diag}\left\{\hat{A}\right\} = \textbf{diag}\{A\}, \tag{7.1}$$

where $\textbf{diag} : \mathcal{S}_+^n \longrightarrow \mathbb{R}^n$ is an operator which maps the diagonal of a matrix to a vector. For any $M \in \mathcal{S}_+^n$, $\|M\|_* = \textbf{tr}\{M\}$. From this and (7.1), we conclude that

$$\left\|\hat{A}\right\|_* = \|A\|_*.$$

This must hold for any arbitrary A which is a solution to Problem 1, so we can rewrite Problem 1 as

$$\begin{aligned} &\hat{A} = M \\ &\text{subject to } \; M \in \mathcal{S}_+^n \\ &\qquad\qquad\quad \|M\|_* = \|A\|_* \\ &\qquad\qquad\quad \mathcal{A}_\Omega\left(M\right) = \mathcal{A}_\Omega\left(A\right). \end{aligned}$$

By the above arguments, $\mathcal{A}_\Omega\left(M\right) = \mathcal{A}_\Omega\left(A\right)$ implies $\|M\|_* = \|A\|_*$ and the equivalence with Problem 3 then immediately follows.

□

Having shown that Problems 2 and 3 are equivalent under our sufficient conditions, we now analyze problem 3. We show that under our sufficient conditions, Problem 3 exactly recovers A. This may seem surprising as Problem 3 has completely removed any penalty on the rank of the returned matrix. Instead, in solving problem 3, we are simply solving a feasibility problem - we seek any matrix $\hat{A}$ which agrees with the observed entries of A.

Our challenge will then to be show that under our sufficient conditions there is only a single matrix in the SPSD cone with entries which match the observed entries of A. We achieve this in two steps. In the first step, we will show that under our sufficient conditions, any matrix which is a solution to Problem 3 must be rank r. This will establish that Problem 3 is equivalent to solving the same problem but constraining the search space to the set of rank r matrices. With this final equivalency established, we call upon the results of chapter 3 to show that the solution to this last problem must be A.

The first step is achieved by the following lemma. The proof is left to the appendix.

Lemma 14. *Let Ω index the observed entries of A such that conditions $A1$ - $A4$ of chapter 3 hold. Then any $\hat{A}$ which is a solution to Problem 3 must be rank r.*

With this intermediate result, we now present the proof of Theorem 12.

Proof of Theorem 12

Proof. By Lemma 14, under conditions $A1$ - $A4$ of chapter 3, Problem 3 is equivalent to the following problem, which we refer to as Problem 4:

$$\begin{aligned} &\hat{A} = M \\ &\text{subject to } \; M \in \mathcal{S}_{+}^{n} \\ &\qquad\qquad\quad \textbf{rank}\{M\} = r \\ &\qquad\qquad\quad \mathcal{A}_{\Omega}(M) = \mathcal{A}_{\Omega}(A). \end{aligned}$$

Problem 4 is a formal statement of the problem considered in chapter 3 - that of recovering a SPSD matrix $\hat{A}$ of known rank subject to agreement between the observed entries of a SPSD matrix A and $\hat{A}$. Theorem 2 of chapter 3 shows there exists an algorithm to find $\hat{A}$ and that $\hat{A} = A$ under conditions $A1$ - $A4$. Therefore, $\hat{A}$ recovered by solving Problem 4 must equal A. Finally, by the series of equivalencies we have established between Problems 1, 3 and 4, we conclude that the under conditions $A1$ - $A4$ of chapter 3 , it must be that $\hat{A}$ recovered by solving Problem 1 is equal to A. □

7.4 Empirical Results for Noisy Matrix Recovery

Algorithm 1 of chapter 3 is an efficient method for completing SPSD matrices in noiseless settings, but it has notable shortcomings for the noisy scenario. In particular, Algorithm 1 estimates a row of the matrix $\hat{C}$ from a single principal submatrix. However, in practice, there may be multiple principal submatrices which can be used for estimating any single row of C. In the noiseless setting this presents no problem, but in a noisy setting, this results in suboptimal estimation of $\hat{C}$ and therefore $\hat{A}$. Second, Algorithm 1 is a single pass algorithm. Given an ordering of principal submatrices, appropriate parts of the full $\hat{C}$ matrix are learned and each is in turn incorporated into the growing estimate of $\hat{C}$ until all rows are learned. However, in practice there may be multiple possible orderings of principal submatrices which meet conditions A1-A4, and an algorithm which completes the matrix in a more global fashion may outperform Algorithm 1.

Motivated by these observations, we perform matrix recovery via the nuclear norm and compare performance to the method we present in chapter 3. We refer to the algorithm in chapter 3 as the Procrustes method of matrix completion, as it relies on Procrustes analysis to align the $\hat{C}_l$ matrices in each iteration.

We compare the performance of these two methods on reconstructing 30 by 30 matrices when we observe 10 by 10 principal submatrices running along the diagonal of each matrix with overlapping 5 by 5 blocks between principal submatrices as shown in Fig. 7.1A. We run simulations when the matrices we seek to recover are incoherent and when they are coherent. We first describe the simulations for recovering incoherent matrices. We randomly generate rank 1 to 5 incoherent matrices by generating a matrix $H \in \mathbb{R}^{n \times r}$, where each entry of H is independently randomly generated from a $\mathcal{N}(0, 1)$ distribution. We then form a SPSD matrix, $A = HH^T$. Fig. 7.1B shows an example of a random matrix generated in this way. To simulate noise we randomly generate a matrix $G \in \mathbb{R}^{n \times n}$, where each entry of G is independently randomly generated from a $\mathcal{N}(0, \sigma_{\text{Noise}})$ distribution. We vary the level of σ_{Noise} between 0 and 1 to generate different amounts of noise. Given G, we form a matrix of SPSD noise as $N = GG^T$ and then form $\tilde{A} = A + N$. Given the observed entries of $\tilde{A}$ in the 10 by 10 principal submatrices sampled, we attempt to recover A via nuclear norm minimization and the Procrustes method. For values of $\sigma_{\text{Noise}} = 0$ (no noise), we attempt recovery via the nuclear norm approach by solving Problem 1 above. When $\sigma_{\text{Noise}} > 0$, we attempt recovery by solving Problem 2 and vary μ between .001 and 1000 in 16 points equally spaced logarithmically. For each simulated matrix recovery problem, we select the value of μ that minimizes error. In practice μ would need to be selected via a heuristic or with cross-validation, but as we seek an estimate of theoretical best performance, we chose to simply sweep the values of μ and report lowest error achieved in this way. Given a simulated matrix A and an estimated matrix $\hat{A}$, we quantify error as $\left\| A - \hat{A} \right\|_F / \left\| A \right\|_F$. For each rank and value of σ_{Noise}, we perform a 100 simulations.

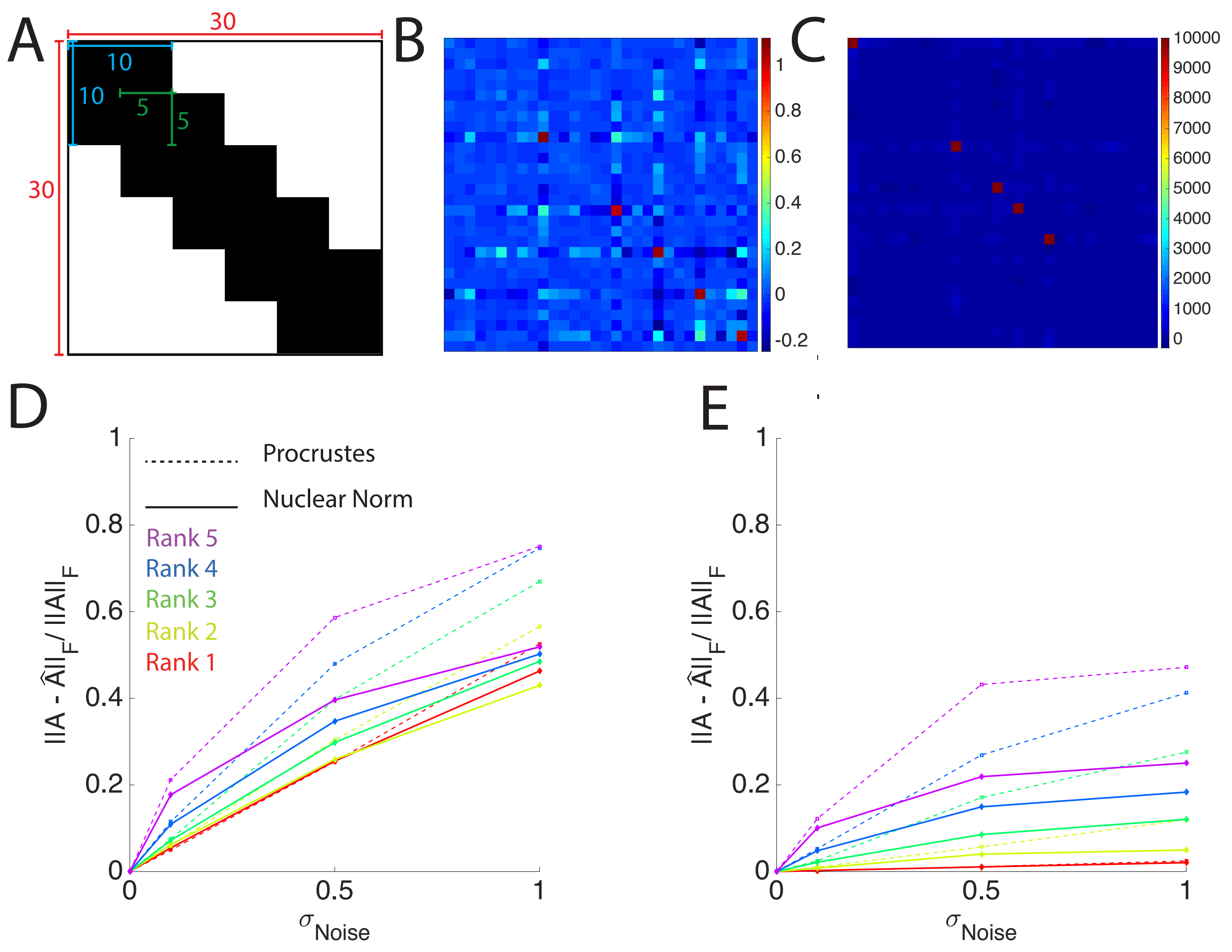


Figure 7.1: Results of matrix recovery simulations. (A) Matrices of size 30 by 30 and varying rank were recovered from entries observed in 10 by 10 submatrices running down the diagonal of the generated matrices. (B) Example of a randomly generated rank 5 incoherent matrix. (C) Example of rank 5 randomly generated coherent matrix. (D) Reconstruction accuracy of incoherent matrices of varying rank for different noise levels. Each point is the average of 100 random simulations. (E) Reconstruction accuracy for coherent matrices. Each point is again the average of 100 random simulations.

Fig. 7.1D presents the results for the recovery of incoherent matrices. As can be seen, the nuclear norm approach uniformly outperforms the Procrustes based method in this simulation. Further, as the rank of the underlying matrix grows, the discrepancy between the two methods widens. This likely reflects the fragility of the Procrustes based method, which is dependent on accurately decomposing each each principal submatrix to recover the individual $\hat{C}_l$ matrices. Importantly, as predicted by theory, both methods are able to exactly recover the underlying matrices in the no noise case.

We also investigated the performance of the two methods when recovering coherent matrices. For each simulation, we generated a coherent matrix by first randomly generating $H \in \mathbb{R}^{n \times r}$ as before. We then randomly selected a single element in each column of H and set it to 100. We did this in a manner so that no two elements in a row of H were set to 100. Fig. 7.1C shows an example of a random matrix generated in this way. We then performed simulations exactly as before. Fig. 7.1E plots average results over 100 random simulations for each rank and value of σ_{Noise}. As can be seen, the nuclear norm based approach continues to outperform the Procrustes based method, and in the no noise simulations, both methods exactly recover the underlying coherent matrices.

Chapter 8

Conclusion and Future Work

Modern techniques for recording neural population activity have made many scientific and clinical advances possible but still face limitations in the number of neurons they can record, the locations of the brain they can be deployed in and in their ability to stably record from a given set of neurons. These limitations have important implications for the practical utility of current BCI systems and constrain the types of scientific questions we can ask. In this thesis, we have presented theory and computational approaches to extend the utility of current recording techniques in the face of these limitations by combining neural population recordings across time and space.

A key component of the work presented in this thesis is the leveraging of structure in neural data. We first leveraged clustering of neural activity in a BCI classification scenario to develop robust classifiers capable of performing self-recalibration to maintain stable performance for up to 31 days. We then turned our attention to low-dimensional structure, which has been found in neural recordings in a wide variety of settings. We presented novel matrix completion theory, applicable to completing a covariance matrix for a population of neurons with low-dimensional neural activity when we can only record from the population in small, overlapping blocks of neurons. To extend our capabilities beyond that of simply completing a covariance matrix, we showed how FA models for a joint population of neurons can be fit to such a set of recordings. The fitted models can then be used to infer the low-dimensional state of the entire population of neurons at any point in time from the subset of the recorded neurons in each block. We developed necessary conditions for fitting FA models in these scenarios and then validated our methods in a BCI regression scenario and when studying the neural basis of learning and communication between brain areas.

A more detailed summary of the contributions of this thesis follows. We conclude by considering future research directions.

8.1 Summary

In chapter 1, we described how we measure neural activity using spike counts and introduced the two types of structure we leverage in this work: clustering and low-dimensional structure. We reviewed limitations in current recording technology which motivate the methods presented in this thesis and discussed how our work on combining neural population recordings differs that from the recent work of others, such as that of Turaga et al. [22], who first used the term stitching to describe the general idea of combining neural population recordings.

In chapter 2, clustered structure in patterns of neural activity in a BCI classification scenario were used as the basis for designing self-recalibrating classifiers. A detailed analysis of how tuning parameters change from day-to-day revealed that tuning parameters on the same electrode tended to move between days in concert, informing decoder design. We presented a fully probabilistic self-recalibrating classifier as well as a simplified version of this decoder which tracked point estimates of tuning parameters. Both classifiers were capable of maintaining stable performance over periods of 26 and 31 days without any supervised retraining.

In chapter 3, we considered the problem of completing a SPSD matrix from a set of its observed principal submatrices. We presented an algorithm and set of sufficient conditions under which exact matrix recovery could be guaranteed. We additionally showed that our sufficient conditions were often necessary, derived an error bound for the problem of recovering partially observed SPSD matrices corrupted by noise and verified our theoretical results through simulations.

In chapter 4, we described how neural recordings could be combined by fitting a single FA model to separate recordings of overlapping populations of neurons. Having fit such an FA model, the low-dimensional state of the population can be inferred in a meaningful way from the neural activity of the neurons in each recording. We use the term “stitching” in this work to specifically refer to combining neural population recordings in this way. We used the theory of chapter 3 to derive necessary conditions for fitting FA models in this scenario and leveraged the algorithm for SPSD matrix completion to combat the problems of local optima that arose from random initializations of EM in the stitching setting.

With chapter 4 providing the foundation, we then applied stitching in three settings. In chapter 5 we returned to the BCI setting, but instead of classification, we considered the problem of designing a self-recalibrating regression algorithm. We used the two-stage algorithm of Sadtler et al. [12] and rendered it self-recalibrating by using stitching to adapt the FA stage of the decoder during normal BCI use. We also presented a method for identifying stable electrodes across days in an unsupervised manner. The resulting adaptive decoder was able to maintain stable BCI performance when simulated electrode failures substantially degraded the performance of a non-recalibrated decoder.

In chapter 6 we examined the utility of stitching for basic science. Using prerecorded data from a single day-learning experiment, we found that the visualized progression of neural state in a simulated neural stitching scenario was qualitatively similar to the visualized progression in a setting where all neurons could be recorded simultaneously. We then performed a series of simulations seeking to quantify this general observation. We simulated scenarios where it was impossible to record from the same set of electrodes for a duration of an experiment. We then estimated low-dimensional state by using only those electrodes common to all recording points and using stitching to combine recordings from all electrodes. We found that when using only the common electrodes, the accuracy of the estimated low-dimensional state degraded with higher rates of electrode turnover. In contrast, low-dimensional state estimated with stitching degraded only slightly as the rate of electrode turnover increased.

In the second half of chapter 6 we presented two methods, reduced rank regression (RRR) and canonical correlation analysis (CCA) for detecting patterns of covariability between two populations of neurons. We explained how each could be potentially useful for detecting subspaces containing neural activity patterns arising from communication between two populations of neurons and could be used in a stitching setting. Using data from populations of neurons simultaneously recorded in V1 and V2, we examined the benefits of stitching when applying RRR and CCA. In our simulations we assumed it was possible to simultaneously record from all V1 neurons but only a limited number of V2 neurons. We then compared the quality of the V1 subspaces found in RRR and CCA when these models were fit in scenarios where we simulated recording from a fixed set of V2 neurons and when we increased the number of V2 neurons sampled during an experiment through stitching. We found that RRR with stitching returned subspaces of V1 neural activity that were better able to predict V2 neural activity that those subspaces returned by RRR fit in the non-stitching simulations. For CCA, we failed to see a benefit to stitching in the simulated scenarios based on the particular dataset we used in our work. We provide evidence that this may be due to small correlations among V2 neurons and fragility of our current stitching algorithm in this particular setting.

Finally, in chapter 7 we returned to the basic question of matrix completion. We proved that the sufficient conditions presented in chapter 3 were also sufficient for the recovery of SPSD matrices via the nuclear norm. This is noteworthy because incoherence, a common assumption in much of the nuclear norm matrix completion literature, was not among the set of sufficient conditions. Additionally, we examined the accuracy of matrices recovered in the presence of noise with the algorithm presented in chapter 3 and with the nuclear norm. We find that in our set of simulations matrices were more accurately recovered with the nuclear norm, suggesting a means of completing SPSD matrices and, more generally, stitching neural recordings together more robustly.

8.2 Future Work

We conclude by briefly commenting on some possible future directions from the work we have presented in this thesis. There are a variety of ways the work can be extended by both developing further theory and application.

As was true with the development of the work to this point, advances in theory and application will benefit from simultaneous exploration, allowing findings in one to inform research directions in the other.

In the area of application, it would be very exciting to test the self-recalibrating classifiers and regression algorithm in an online setting. While the offline results we present are very encouraging, a closed loop setting where a subject can adapt their neural signals to account for instabilities in day-to-day recordings raises an important question. Could adapting the decoder essentially produce a "moving target" for a subject which hinders BCI performance or would learning be improved as the decoder essentially meets the subject "in the middle" reducing the amount of learning a subject must do from day-to-day? Both possibilities are intriguing and online experiments will be required to differentiate the two. In fact, we plan to test the adaptive regression algorithm in an online setting in the immediate future.

It would also be exciting to study learning across multiple days with stitching. For reasons mentioned in chapter 6, this has been difficult to do with modern electrophysiological techniques up to this point in time. This is a good example of where application and theory can develop together. Requiring a dataset where we could assume recordings were stable for validation purposes, we have presented simulated results for a single-day learning experiment in this thesis. The success of stitching on this dataset is important, but across day learning may differ from within day learning in important ways. For example, there is increasing evidence that sleep plays an important role in learning motor skills (e.g., [124,125]). Could this cause neural dynamics to evolve across days in a manner which is fundamentally different than it evolves within a day? It is also likely that learning will change the underlying low-dimensional state distribution (i.e., the distribution on l_t in an FA model) between days. It could also change the mapping between low-dimensional state and firing rate patterns (i.e., the C matrix of an FA model). Our stitching algorithms implicitly assume these are stationary. It is encouraging that in our single day experiments, where these non-stationarities are likely to be at least mildly present, we are able to still successfully infer changes in neural state with learning, but perhaps our algorithms could be generalized to more robustly handle these types of non-stationarities.

Regarding the last proposed application area, there are at least two ways the methods we present to study communication between brain areas can be extended. First, in chapter 6 we propose that RRR and CCA may return subspaces containing neural activity patterns arising from communication between populations of neurons. However, this is a claim that is best validated through experiments which allow for the quantification of behavioral correlates along with changes in neural activity patterns. For example, an experiment which found changes in reaction time or task success when neural activity lied within or outside of a putative "communication" subspace between two areas would provide increased motivation for the identification of such subspaces. Depending on the areas involved, this type of validation can be done without stitching. In a second line of work, the use of stitching with CCA should continue to be evaluated. For the dataset we analyze, it seemed that accounting for the covariance structure between paris of V2 neurons minimally affected the quality of the returned V1 subspace. It seems this finding may be specific to the dataset we analyze. For example, if more V2 neurons were sampled or sampled more densely, this could change. Thus, the use of CCA with stitching and different validation datasets should be investigated.

Finally, the finding that the nuclear norm can recover SPSD matrices in the noiseless setting under our set of sufficient conditions suggests fruitful directions for future work. First, we show empirically that matrices recovered via the nuclear norm are recovered more accurately than those with the algorithm presented in chapter 3. While theory pertaining to matrix recovery in the noisy setting typically involves sophisticated proofs, establishing an error bound in the noisy setting would add rigor to our empirical findings. We note that the reconstruction of SPSD matrices arises in multiple areas in machine learning and science and developing robust methods of recovering such matrices is of general interest. In addition to this theoretical work, the algorithm of chapter 3 could be replaced with a method based on the nuclear norm to initialize EM for fitting FA models. This could be done in two ways. In the first, just as individual portions of the low-rank covariance structure are currently estimated (that is the $\hat{C}_l$ matrices of Algorithm 1 in chapter 3) and then stitched together, these could be stitched together be forming blocks of $\hat{C}_l\hat{C}_l^T$ submatrices and completing the full $\hat{C}\hat{C}^T$ matrix via the nuclear norm. Alternatively, and perhaps more elegantly, the low-rank covariance structure for an FA model could be estimated by solving a nuclear norm problem which measures error on all entries of the covariance matrix which can be computed entry-wise with the exception of the diagonal entries, as the diagonal entries will be corrupted by the private noise of individual neurons. Both of these methods could be investigated theoretically and their use to improve the robustness of stitching investigated.

Appendix A

Appendix for A Self-Recalibrating Classifier for Brain Computer Interface

A.1 Additional Methods

A.1.1 The EM Algorithm

This section presents the expectation maximization (EM) algorithm used to learn the parameters of the SR classifier.

The E-Step

The E-Step computes the distribution of the unobserved (or latent) variables of the model given all of the observed data in $\mathcal{D}_{\text{Train}}$ and the parameter estimates from the previous M-Step. For the SR classifier, the latent variables are the base values, b_d, for each day in $\mathcal{D}_{\text{Train}}$, and the observed data are the neural activity, $x_{d,t}$, and reach directions, $y_{d,t}$, for each trial during each day in $\mathcal{D}_{\text{Train}}$. Mathematically, we write the probabilities of the latent variables given the observed variables as $P(\{b_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$, where we group all of the neural activity for day d into the matrix $X_d \in \mathbb{R}^{E\times T_d}$ and all of the reach directions into the row vector $Y_d \in \mathbb{R}^{1\times T_d}$, where T_d is the total number of trials on day d.

We desire to derive an expression for $P(\{b_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$ and start by using the independence of reach directions, $y_{d,t}$, and base values, b_d, across days to decompose this distribution as

$$P(\{b_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}}) = \prod_{d\in\mathcal{D}_{\text{Train}}} P(b_d|X_d, Y_d). \tag{A.1}$$

We further decompose the right hand side of eq. A.1 by recognizing that base values conditioned on spike counts and reach direction are independent from one another and write

$$P(b_d|X_d, Y_d) = \prod_{e=1}^{E} P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d), \tag{A.2}$$

where the notation $\{x_{d,e,t}\}_{t=1}^{T_d}$ indicates the set of spike counts for electrode e for all trials on day d.

To compute $P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d)$, we note that by Bayes' rule we have:

$$\begin{aligned} P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d) &\propto P(\{x_{d,e,t}\}_{t=1}^{T_d}|b_{d,e}, Y_d)P(b_{d,e}|Y_d) \\ &= P(\{x_{d,e,t}\}_{t=1}^{T_d}|b_{d,e}, Y_d)P(b_{d,e}), \end{aligned} \tag{A.3}$$

where we have used the independence of base value and reach directions in moving to the second line above. We then use the independence of neural activity conditioned on base value to decompose $P(\{x_{d,e,t}\}_{t=1}^{T_d}|b_{d,e}, Y_d)$ from the right hand side of eq. A.3 as

$$P(\{x_{d,e,t}\}_{t=1}^{T_d}|b_{d,e}, Y_d) = \prod_{t=1}^{T_d} P(x_{d,e,t}|y_{d,t}, b_{d,e}). \tag{A.4}$$

We are now in a position to derive the distribution $P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d)$ for each day in a straight forward manner. eqs. A.1 and A.2 can then be used to arrive at the final distribution specifying $P(\{b_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$. We first recognize that $P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d)$ must be a Gaussian distribution, for as shown by eq. A.3, it is proportional to the product of two Gaussian distributions. The first of these distributions is $P(b_{d,e})$, which is defined as a Gaussian by eq. 2.3 in the main text with mean m_e and variance s_e, and the second is $P(\{x_{d,e,t}\}_{t=1}^{T_d}|Y_d, b_{d,e})$, which can be viewed as a Gaussian distribution over $b_{d,e}$. To see this note that viewed as function of the base value, $b_{d,e}$, for a fixed spike count, $x_{d,e,t}$, and reach directions, $y_{d,t}$, each term on the right hand side of eq. A.4, $P(x_{d,e,t}|y_{d,t}, b_{d,e})$, is a Gaussian distribution with mean $x_{d,e,t} - o_{e,y_{d,t}}$ and variance $v_{e,y_{d,t}}$, where the trial label $y_{d,t}$ appears as a subscript to indicate these are the offsets and variances for electrode e and reach direction j when $j = y_{d,t}$. Thus, the right hand side of eq. A.4 is a product of univariate Gaussians, and the left hand side of this equation must also be a Gaussian.

Having identified $P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d)$ as Gaussian it remains to specify its mean and variance. We refer to the mean of $P(b_{d,e}|\{x_{d,e,t}\}_{t=1}^{T_d}, Y_d)$ as $\langle b_{d,e}\rangle$ and variance as $\text{Var}(b_{d,e})$. The standard formula for the product of Gaussians can be used to find $\langle b_{d,e}\rangle$ and $\text{Var}(b_{d,e})$. We first state the equation for the variance, which is

$$\text{Var}(b_{d,e}) = \left(\sum_{t=1}^{T_d} \frac{1}{v_{e,y_{d,t}}} + \frac{1}{s_e}\right)^{-1}. \tag{A.5}$$

The mean, $\langle b_{d,e}\rangle$, can then be shown to be

$$\langle b_{d,e}\rangle = \text{Var}(b_{d,e}) \left(\sum_{t=1}^{T_d} \frac{x_{d,e,t} - o_{e,y_{d,t}}}{v_{e,y_{d,t}}} + \frac{m_e}{s_e}\right). \tag{A.6}$$

The M-Step

In the M-step, we desire to maximize

$$\mathcal{E}(\Omega) = \mathbb{E}\left[\log P(\{X_d\}_{d\in\mathcal{D}_{\text{Train}}}, \{Y_d\}_{d\in\mathcal{D}_{\text{Train}}}|\Omega)\right], \tag{A.7}$$

with respect to Ω, which is made up of the m_e, s_e, $o_{e,j}$ and $v_{e,j}$ parameters. The expectation is taken with respect to the distribution $P(\{b_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$ found in the E-step given in eq. A.1. Maximizing $\mathcal{E}(\Omega)$ with respect to the m_e and s_e parameters yields

$$m_e = \frac{\sum_{d\in\mathcal{D}_{\text{Train}}} \langle b_{d,e}\rangle}{|\mathcal{D}_{\text{Train}}|} \tag{A.8}$$

$$s_e = \frac{\sum_{d\in\mathcal{D}_{\text{Train}}} \left(\text{Var}(b_{d,e}) + (\langle b_{d,e}\rangle - m_e)^2\right)}{|\mathcal{D}_{\text{Train}}|}, \tag{A.9}$$

where $|\mathcal{D}_{\text{Train}}|$ is the number of training days. The updates for the $o_{e,j}$ parameters are calculated as

$$o_{e,j} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} \sum_{t: y_{d,t}=j} (x_{d,e,t} - \langle b_{d,e} \rangle)}{\sum_{d \in \mathcal{D}_{\text{Train}}} n_{d,j}}, \tag{A.10}$$

where $n_{d,j}$ is the number of trials on day d with reaches in direction j, and the notation $t : y_{d,t} = j$ indexes the trials on day d in which the subject reached in direction j. Finally, values for the $v_{e,j}$ parameters are calculated as

$$v_{e,j} = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} \sum_{t: y_{d,t}=j} \left(\text{Var}(b_{d,e}) + (\langle b_{d,e} \rangle + o_{e,j} - x_{d,e,t})^2\right)}{\sum_{d \in \mathcal{D}_{\text{Train}}} n_{d,j}}. \tag{A.11}$$

Note that in practice we do not iterate estimates for m_e. This is because there is a degeneracy in the parameters such that multiple settings of the m_e and $o_{e,j}$ parameters result in the same data likelihood. Specifically, it can be shown that the set of offset parameters, $o_{e,j}$, can be chosen with respect to any given set of m_e parameters such that the value of $\mathcal{E}(\Omega)$ is unchanged. While we omit a formal proof, the intuition that underlies this is quite simple and found in eq. 2.5 in the main text. Critically, what determines the likelihood of data, $P(\{X_d\}_{d \in \mathcal{D}_{\text{Train}}}, \{Y_d\}_{d \in \mathcal{D}_{\text{Train}}} | \Omega)$, in eq. A.7 is the set of electrode-class means, $\mu_{d,e,j}$, for all classes, electrodes and days. For a given electrode, eq. 2.5 shows that the class mean for electrode e and class j on day d is formed from a linear combination of the base value for the electrode and day, $b_{d,e}$, and the offset for the electrode and class, $o_{e,j}$. Importantly, we can subtract a given quantity from the base value, $b_{d,e}$, and add that same quantity to all offsets, $o_{e,j}$, for that electrode and leave class means, $\mu_{d,e,j}$, for that electrode unchanged. What this immediately implies is that we can also subtract a given quantity from m_e, which is the mean of the distribution for daily base values for the electrode, and add that same quantity to all the offset parameters, $o_{e,j}$, for that electrode, and leave the distribution over class-electrode means, $\mu_{d,e,j}$, unchanged. This allows flexibility in choosing values for the m_e parameters, and we chose to set their values equal to the values for the $b_{d,e,0}$ parameters we learn for the SR_S classifiers given by eq. 2.21 in the main text.

Initializing the EM algorithm

For the first E-step, there will be no parameter estimates from a previous M-step with which to calculate the quantities above with. Therefore, initial values must be provided, and we leverage the training algorithm for the SR_S classifier. We estimate values for the m_e parameters using eq. 2.21 in the main text. Initial values for the $o_{e,j}$ parameters are estimated with eq. 2.22, and initial values for the $v_{e,j}$ parameters are calculated with eq. 2.23 from the main text. The SR_S classifier has no s_e parameters, and we form an initial estimate of these as

$$\hat{s}_e = \frac{\sum_{d \in \mathcal{D}_{\text{Train}}} (\hat{\mu}_{d,e} - \hat{m}_e)^2}{|\mathcal{D}_{\text{Train}}| - 1}, \tag{A.12}$$

where $\hat{\mu}_{d,e}$ is calculated from eq. 2.19 from the main text.

Calculating the Log Likelihood of the Observed Data: A Stopping Criterion for EM

During each E-step of the EM algorithm, the likelihood of the observed data, $l(\Omega) = P(\{X_d\}_{d \in \mathcal{D}_{\text{Train}}}, \{Y_d\}_{d \in \mathcal{D}_{\text{Train}}} | \Omega)$ can be calculated. For numerical reasons it is not $P(\{X_d\}_{d \in \mathcal{D}_{\text{Train}}}, \{Y_d\}_{d \in \mathcal{D}_{\text{Train}}})$ but $l'(\Omega) = \log l(\Omega)$ that is normally calculated. It is a property of the EM algorithm that $l(\Omega)$ will converge, and therefore, $l'(\Omega)$ must also converge, and monitoring $l'(\Omega)$ provides a convenient stopping criteria. In this work, we iterate the EM algorithm until the difference in $l'(\Omega)$ between two successive iterations fails to increase by a value of less than 10^{-10} or the EM algorithm runs for 10,000 iterations.

A.1.2 Calculating $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1})$ for Flagging Erratically Behaving Electrodes

In this section we show how to compute $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1})$, which is useful for flagging erratically behaving electrodes. We calculate $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1})$ as

$$\begin{aligned} P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}) &= \sum_{j=1}^{J} P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)P(y_{d,t}=j|\{x_{d,s}\}_{s=1}^{t-1}) \\ &= \sum_{j=1}^{J} P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)P(y_{d,t}=j), \end{aligned} \tag{A.13}$$

where we have used the fact that reach direction is independent of neural activity to move to the second line above. We specify $P(y_{d,t}=j)$ to be $1/J$ in section 2.4 of the main text, and we calculate $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)$ as

$$\begin{aligned} P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j) &= \int P(x_{d,e,t}|b_{d,e}, \{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)P(b_{d,e}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j)db_{d,e} \\ &= \int P(x_{d,e,t}|b_{d,e}, y_{d,t}=j)P(b_{d,e}|\{x_{d,s}\}_{s=1}^{t-1})db_{d,e} \end{aligned} \tag{A.14}$$

where we have leveraged the independence of $x_{d,e,t}$ from previous neural activity given $b_{d,e}$ to simplify the first term of the integrand and the independence of $b_{d,e}$ from $y_{d,t}$ conditioned on $\{x_{d,s}\}_{s=1}^{t-1}$ to simplify the second term. $P(x_{d,e,t}|b_{d,e}, y_{d,t}=j)$, is obtained from eq. 2.6 from the main text and is a Gaussian with mean $o_{e,j}+b_{d,e}$ and variance $v_{e,j}$. The second term can be obtained from a marginalization of the distribution for $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$ given by eq. 2.13 in the main text. This marginalization is easily performed since $P(b_d|\{x_{d,s}\}_{s=1}^{t-1})$ is a multivariate Gaussian and we refer to the mean and variance of $P(b_{d,e}|\{x_{d,s}\}_{s=1}^{t-1})$ as $m_{e,t-1}$ and $\sigma^2_{e,t-1}$, respectively. The right hand side of eq. A.14 is composed of two terms and is a convolution of two Gaussians, so it must be that

$$x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1}, y_{d,t}=\ j\ \sim \mathcal{N}\left(o_{e,j}+m_{e,t-1}, v_{e,j}+\sigma^2_{e,t-1}\right). \tag{A.15}$$

Therefore, $P(x_{d,e,t}|\{x_{d,s}\}_{s=1}^{t-1})$, calculated in eq. A.13, is a mixture of Gaussians with means and variance given by eq. A.15.

A.1.3 Proof for the Consistency of Base Value Estimates Produced by the SR_S Classifier

In this section we show that as the number of trials on a decoding day increases, the estimate for the base value of an electrode, $\hat{b}_{d,e,t}$, produced by the SR_S classifier will approach the true mean of the distribution of spike counts for that electrode and day.

We start by noting that $n_{d,t}$ in eq. 2.17 in the main text can be written as

$$n_{d,t} = n_{d,0} + t, \tag{A.16}$$

and that $n_{d,t-1}\hat{b}_{d,e,t-1}$ from eq. 2.18 can be written as

$$n_{d,t-1}\hat{b}_{d,e,t-1} = n_{d,0}b_{d,e,0} + \sum_{s=1}^{t-1} x_{d,e,s}. \tag{A.17}$$

Taken together, this allows us to rewrite eq. 2.18 in a non-recursive form as

$$\hat{b}_{d,e,t} = \frac{n_{d,0}b_{d,e,0} + \sum_{s=1}^{t} x_{d,e,s}}{n_{d,0}+t} = \frac{n_{d,0}b_{d,e,0}}{n_{d,0}+t} + \frac{\sum_{s=1}^{t} x_{d,e,s}}{n_{d,0}+t}. \tag{A.18}$$

If $n_{d,0}$ is finite, as t goes to infinity, the first term in eq. A.18 goes to 0, and the second term will approach $\frac{\sum_{s=1}^{t} x_{d,e,s}}{t}$, which is an expression for the empirical mean for the day. It then follows by the law of large numbers that $\frac{\sum_{s=1}^{t} x_{d,e,s}}{t}$ will approach the true mean of the distribution of spike counts for electrode e on day d.

A.1.4 A Comparison of the SR and SR_S Classifiers

The decoding algorithm presented for the SR classifier seeks to classify trial t using the posterior $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ found with iterative application of *eqs.* 2.8, 2.9, 2.12, and 2.13 from the main text. These equations admit an efficient recursive decoding procedure but obscure the meaning of $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$. To aid understanding, we decompose $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ as

$$P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t}) = \int P(y_{d,t}|x_{d,t}, b_d)P(b_d|\{x_{d,s}\}_{s=1}^{t})db_d, \tag{A.19}$$

where we have used the independence of reach direction from previous neural activity given b_d to write $P(y_{d,t}|x_{d,t}, \{x_{d,s}\}_{s=1}^{t-1}, b_d,)$ as $P(y_{d,t}|x_{d,t}, b_d)$.

Eq. A.19 indicates that we can intuitively conceptualize forming $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ through a two-step process. First, a posterior distribution of daily base values using all of the feature vectors collected for the day, $P(b_d|\{x_{d,s}\}_{s=1}^{t})$, is formed. Second $P(y_{d,t}|\{x_{d,s}\}_{s=1}^{t})$ is calculated as a continuously weighted average of an infinite number of $P(y_{d,t}|x_{d,t}, b_d)$ terms, where each term is associated with a unique value of b_d. We can understand this averaging as accounting for the fact that we don't know b_d with certainty, and instead we allow values of b_d with higher probability to influence our final class decision more and those with less probability to influence our final decision less.

In principle, this is a fully probabilistic means of classification. However, the approximation repeated for each iteration of the decoding algorithm presented with eq. 2.13 in the main text means that $P(b_d|\{x_{d,s}\}_{s=1}^{t})$ in eq. A.19 is also approximate. This approximation produces an algorithm for efficiently classifying reach direction while also modeling uncertainty in b_d, but alternative tradeoffs can be made.

In particular, we can forego modeling uncertainty in b_d and instead decode with a single point estimate for b_d formed for each trial. The SR_S classifier forms this point estimate, $\hat{b}_{d,t}$, for each trial with eqs. 2.17 and 2.18 in the main text and then calculates posterior probabilities for reach direction as $P(y_{d,t}|x_{d,t}, b_d = \hat{b}_{d,t})$. This is equivalent to replacing the distribution $P(b_d|\{x_{d,s}\}_{s=1}^{t})$ in eq. A.19 with a distribution which has infinite density at the point $b_d = \hat{b}_{d,t}$ and is zero everywhere else.

A.2 Additional Results

A.2.1 Tuning Parameter Drift: Results Obtained with Alternative Kernel Widths

In section 2.3.1, we present a characterization of neural signal drift by first convolving spikes with a Gaussian kernel with a width of 100 trials to estimate trial-to-trial tuning parameter values and then examining statistics calculated on these estimated values. We show that between-day drift is more substantial than within-day drift, that tuning parameters on the same electrode drift in a correlated fashion across days and that tuning parameters drift in a constrained manner from day to day. Here we reproduce the essential panels of Figs. 2.5 and 2.6 from the main text using different kernel widths to provide evidence that these findings are not sensitive to the width of kernel used.

Figs. A.1 and A.2 present these general results. From top to bottom, each row of each figure shows results obtained with kernel widths of 25, 50, 100 and 200 trials. Panels in the left column of each figure show the distribution of the standard deviation of tuning parameter values within and between days. The distributions of correlation coefficients for the day-to-day drift in tuning parameters on the same and different electrodes are shown in the middle columns of each figure. Finally, panels in the right column of each figure show the average values of unsigned percent change of mean tuning parameter values from their value on day 1. At first glance, the displayed results may appear identical for all kernel widths; however, this is not a plotting error and careful inspection will reveal subtle differences.

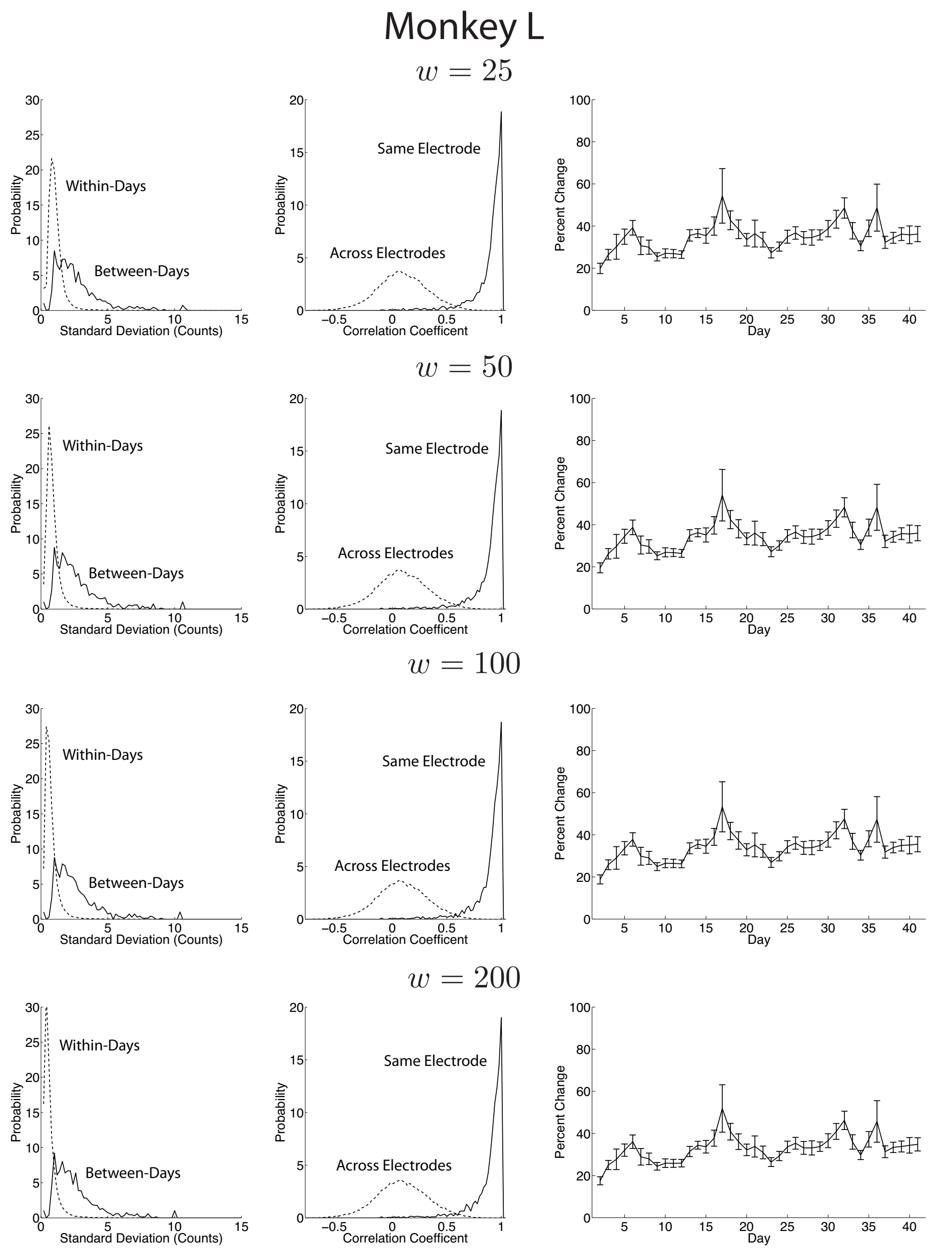


Figure A.1: This figure reproduces the analysis on data from Monkey L to produce panels B (left column), D (middle column) and E (right column) from Fig. 2.5 from the main text when kernels with widths of 25, 50, 100 and 200 trials were used when convolving spikes with a Gaussian kernel to estimate trial-to-trial tuning parameters. A small number (.08% for w = 25, 50, 100 and .04% for w = 200) of within-day correlation coefficients are outside the range of the plots in the middle column. Three electrodes were withheld when calculating average unsigned percent change in tuning parameters from their value on day 1 to produce the right most panel in each column because the value of one or more of the tuning parameters for these electrodes on day 1 was very small. See text of section 2.2.2 for details.

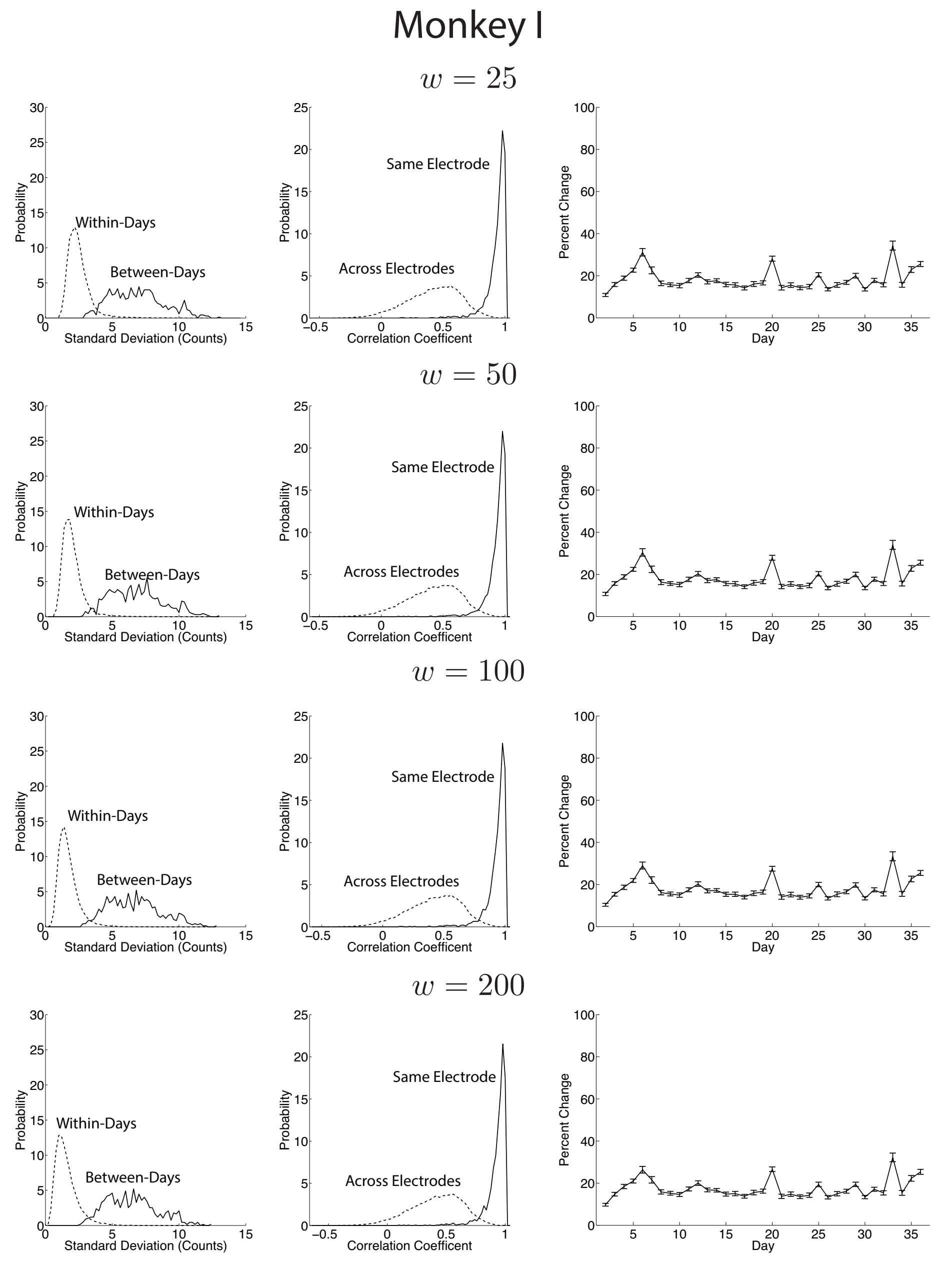


Figure A.2: This figure reproduces the analysis on data from monkey I to produce panels A (left column), B (middle column) and C (right column) from Fig. 2.6 from the main text when kernels with widths of 25, 50, 100 and 200 trials were used when convolving spikes with a Gaussian kernel to estimate trial-to-trial tuning parameters. For monkey I, no electrodes were withheld in the analysis to produce the panels in the right column.

A.2.2 Drift in Modulation Depth

We have performed a preliminary characterization in the modulation depth of electrodes over time. Given an electrode with J tuning parameters (the $\mu_{d,e,j,t}$ parameters in section 2.3.1), we calculate the modulation depth by subtracting the value of the smallest tuning parameter from that of the largest. We calculate the modulation depth for an electrode on a trial-by-trial basis across all days from the tuning parameter estimates obtained in section 2.3.1. After obtaining estimates of modulation depth across time in this manner, we then measure the standard deviation of the modulation depth for an electrode on each day in the dataset. This set of values quantifies the within-day variance in modulation depth for that electrode. We also take the average modulation depth on each day and measure the standard deviation in these average daily values to quantify the between-day variation. We do this for all electrodes and present the results in Fig. A.3 below. There is evidence for changes of both within and between-day changes in modulation depth. As with the main findings on tuning parameter drift, results are presented when kernel widths of 25, 50, 100 and 200 trials were used to estimate the underlying tuning parameters. For monkey L, the between-day changes appear larger than within-day changes for all kernel widths that were used to estimate the original parameter values. For monkey I, between-day changes appear substantially larger than within-day changes only for larger kernel widths.

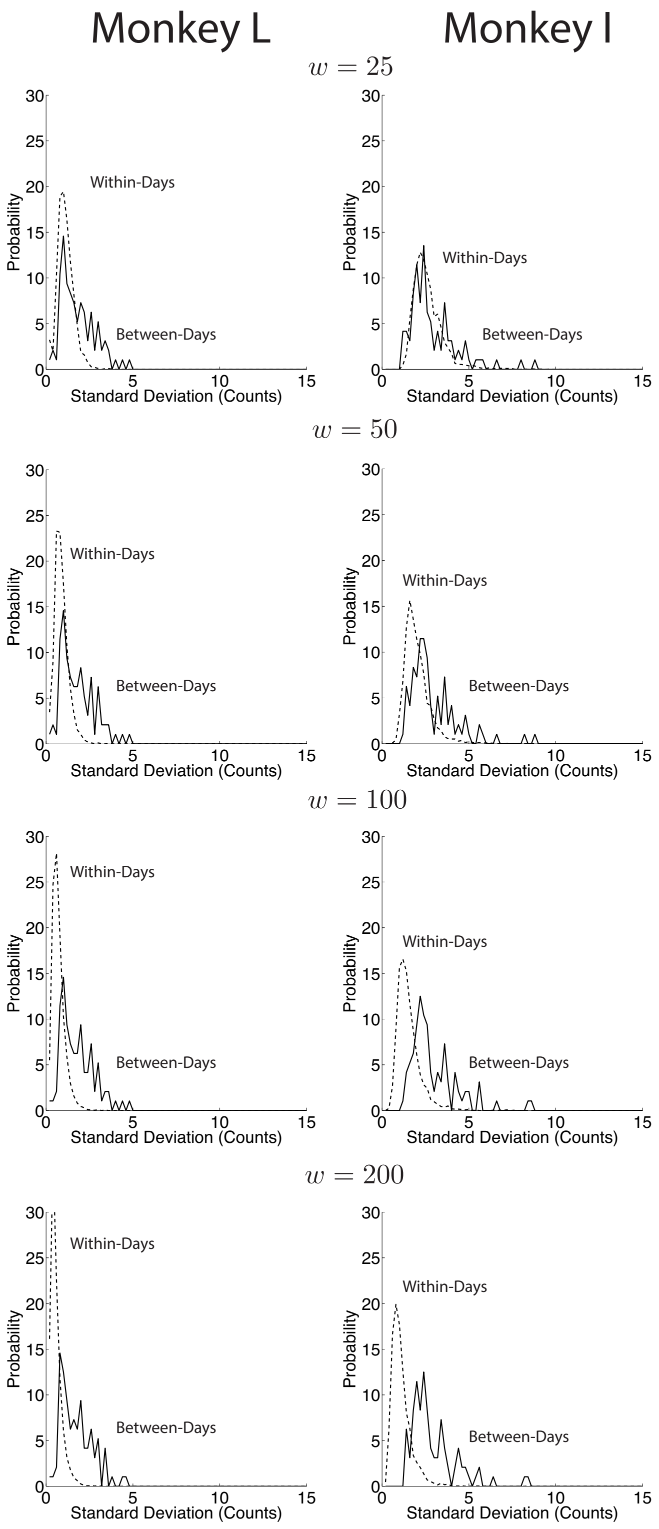


Figure A.3: Histograms of the standard deviation of within and between day modulation depth values across all electrodes for both monkeys. For completeness, results derived from tuning parameters estimated with Gaussian smoothing with kernel widths of 25, 50, 100 and 200 trials are shown.

Appendix B

Appendix for Deterministic Symmetric Positive Semidefinite Matrix Completion

B.1 Proofs of the Main Results

B.1.1 Proof of Theorem 1

Proof. The theorem is a restatement of lemma 20, which we state and prove in the technical results section below. For clarity, here we formally establish the correspondence between the theorem and lemma 20.

First, note that by assumption $\mathbf{rank}\{A\} > 0$. Let $\Omega_1 = \rho_1 \times \rho_1$ and $\Omega_2 = \rho_2 \times \rho_2$ be the two index sets in the theorem. By assumption we have $\rho_1 \times \rho_1 \cup \rho_2 \times \rho_2 = \Omega$ and $\Omega \neq [n] \times [n]$. If $A1$ is not met, then $\rho_1 \cup \rho_2 \neq [n]$, and from lemma 20 we can conclude recovery of A is impossible.

If $\rho_1 \cup \rho_2 = [n]$, but $A2$ is not met then $\iota_2 = |\rho_1 \cap \rho_2| < r$ so it must be that $\mathbf{rank}\{A(\iota_2, \iota_2)\} < r$. Further, by assumption $\mathbf{rank}\{A(\rho_1, \rho_1)\} = \mathbf{rank}\{A(\rho_2 \setminus \iota_2, \rho_2 \setminus \iota_2)\} = r$. By lemma 20, we can again conclude that matrix recovery is impossible. Finally, if $A3$ is not met, then again $\mathbf{rank}\{A(\iota_2, \iota_2)\} < r$ and we can conclude matrix recovery is impossible.

□

B.1.2 Proof of Theorem 2

Proof. Our proof technique will be to show that at the end of step 2 of the algorithm, the algorithm has learned a $\hat{C}$ such that $\hat{A} = \hat{C}\hat{C}^T = A$ in step 3 of the algorithm.

We establish the result by induction, using $k = 1$ as the base case. If k = 1, algorithm 1 consists of running step 1 followed immediately by step 3. The matrix $A(\rho_{\tau_1}, \rho_{\tau_1})$ is a principal submatrix of A by assumption $A2$ and must therefore be SPSD. Therefore, we can decompose it with a real eigendecomposition, and $\hat{C}(\rho_{\tau_1}, :) = E_{\tau_1}\Lambda_{\tau_1}^{1/2}$ in step 1 of the algorithm, so it must be that $\hat{C}(\rho_{\tau_1}, :) \in \mathbb{R}^{|\rho_{\tau_1}| \times r}$ and

$$\hat{A}(\rho_{\tau_1}, \rho_{\tau_1}) = \hat{C}(\rho_{\tau_1}, :)\hat{C}(\rho_{\tau_1}, :)^T = E_{\tau_1}\Lambda_{\tau_1}E_{\tau_1}^T = A(\rho_{\tau_1}, \rho_{\tau_1}).$$

If $k > 1$, there are $k - 1$ iterations of step 2 between step 1 and step 3 of the algorithm. Let $l \in \{2, \ldots, k\}$ indicate[1] what iteration the algorithm is on in step 2. Define $\mathcal{I}_l = \cup_{i=1}^{l}\rho_{\tau_i}$. Assume at the start of the iteration of step 2 for a given l, the algorithm has previously assigned values to $\hat{C}(\mathcal{I}_{l-1}, :) \in \mathbb{R}^{|\mathcal{I}_{l-1}| \times r}$ so that

$$\hat{C}(\mathcal{I}_{l-1}, :)\hat{C}(\mathcal{I}_{l-1}, :)^T = \hat{A}(\mathcal{I}_{l-1}, \mathcal{I}_{l-1}) = A(\mathcal{I}_{l-1}, \mathcal{I}_{l-1}).$$

[1]By this indexing notation, we count iterations of step 2 starting with the number 2. For example, if step two runs twice, then for the first iteration of step 2, $l = 2$, and for the second $l = 3$.

Above we have shown this is true for $l = 2$. For part 2a of each iteration, there is again by assumption no noise, so $\tilde{A}(\rho_{\tau_l}, \rho_{\tau_l}) = A(\rho_{\tau_l}, \rho_{\tau_l})$. $A(\rho_{\tau_l}, \rho_{\tau_l})$ again indexes a principal submatrix of A by assumption $A2$, so its eigendecomposition with real eigenvalues exists and we again have $\hat{C}_l = E_{\tau_l}\Lambda_{\tau_l}^{1/2}$, $\hat{C}_l \in \mathbb{R}^{|\rho_{\tau_l}| \times r}$ for $\hat{C}_l$ in step 2a, so

$$\hat{C}_l \hat{C}_l^T = E_{\tau_l}\Lambda_{\tau_l}E_{\tau_l}^T = A(\rho_l, \rho_l).$$

We now use lemma 19 to establish that after steps 2b and c, we will have a $\hat{C}(\mathcal{I}_l, :) \in \mathbb{R}^{|\mathcal{I}_l| \times r}$ such that $\hat{C}(\mathcal{I}_l, :)\hat{C}(\mathcal{I}_l, :)^T = A(\mathcal{I}_l, , \mathcal{I}_l,)$.

To establish that this lemma applies, we need to first establish the correspondence between the variables of algorithm 1 and those used in the statement of the lemma. Specifically, $A(\mathcal{I}_l, \mathcal{I}_l)$ is the matrix to be reconstructed from two of it's principal submatrices $A(\mathcal{I}_{l-1}, \mathcal{I}_{l-1})$ and $A(\rho_{\tau_l}, \rho_{\tau_l})$. We can verify that as required by the lemma $A(\rho_{\tau_l} \cup \mathcal{I}_{l-1}, \rho_{\tau_l} \cup \mathcal{I}_{l-1}) = A(\mathcal{I}_l, \mathcal{I}_l)$. For these two index sets, we have also already established that the algorithm's variables $\hat{C}(\mathcal{I}_{l-1}, :)$ and $\hat{C}_l$ are such that $\hat{C}(\mathcal{I}_{l-1}, :) \in \mathbb{R}^{|\mathcal{I}_{l-1}| \times r}$, $\hat{C}_l \in \mathbb{R}^{|\rho_{\tau_l}| \times r}$, $\hat{C}(\mathcal{I}_{l-1}, :)\hat{C}(\mathcal{I}_{l-1}, :)^T = A(\mathcal{I}_{l-1}, \mathcal{I}_{l-1})$ and $\hat{C}_l\hat{C}_l^T = A(\rho_{\tau_l}, \rho_{\tau_l})$.

Continuing to establish the correspondence between variables in the algorithm and the lemma, we note that the algorithm's variable $\iota_l = \rho_{\tau_l} \cap \left(\cup_{j=1,\dots,l-1}\rho_{\tau_j}\right) = \rho_{\tau_l} \cap \mathcal{I}_{l-1}$. Further, it can be verified that the variables in the algorithm ι_l and ϕ_l index the rows of $\hat{C}$ and $\hat{C}_l$ such that $\hat{C}(\iota_l, :)\hat{C}(\iota_l, :)^T = A(\iota_l, \iota_l) = \hat{C}_l(\phi_l, :)\hat{C}_l(\phi_l, :)^T$, and it can also be verified that algorithm's variable η_l is defined so that $A(\rho_{\tau_l} \setminus \iota_l, \rho_{\tau_l} \setminus \iota_l) = \hat{C}_l(\eta_l, :)\hat{C}_l(\eta_l, :)^T$.

By assumptions $A3$ and $A4$, **rank**$\{A(\iota_l, \iota_l)\} = r$. This establishes that **rank**$\{A(\mathcal{I}_l, \mathcal{I}_l)\} = r > 0$ as required by the lemma. All the lemmas conditions are then met, and it then follows from the lemma that there must be a unique orthogonal W such that $\hat{C}(\iota_l, :) = \hat{C}_l(\phi_l, :)W$. However, if W is unique and establishes the equality, this must be the $\hat{W}_l$ learned in part b of step 2 of the algorithm. Further, we just established the correspondence between the variables in the algorithm and that of the lemma, so in part c of the step 2, we can conclude the algorithm produces a $\hat{C}(\mathcal{I}_l, :)$ such that $A(\mathcal{I}_l, \mathcal{I}_l) = \hat{C}(\mathcal{I}_l, :)\hat{C}(\mathcal{I}_l, :)^T$, which completes the inductive part of the proof.

All that remains of the proof is to show after processing all of the principal submatrices indexed by $\Omega_1, \dots, \Omega_k$, in steps 1 and 2 of the algorithm, step 3 will exactly recover A. If $k = 1$, the algorithm consists of step 1 followed immediately by step 3. In step 1 we have shown that it will assign values to the rows of $\hat{C}(\mathcal{I}_1, :)$ such that $\hat{C}(\mathcal{I}_1, :)\hat{C}(\mathcal{I}_1, :)^T = A(\mathcal{I}_1, \mathcal{I}_1)$. For $k > 2$, the algorithm will perform $k - 1$ iterations of step 2 before step 3. By the inductive part of the proof just finished, we can conclude after these $k - 1$ iterations, the algorithm will have assigned values to the rows of $\hat{C}(\mathcal{I}_k, :)$ such that $\hat{C}(\mathcal{I}_k, :)\hat{C}(\mathcal{I}_k, :)^T = A(\mathcal{I}_k, \mathcal{I}_k)$. However, by assumptions $A1$ and $A2$, $\mathcal{I}_k = \cup_{l=1}^k \rho_{\tau_l} = [n]$, showing that all the rows of $\hat{C}$ have been learned and therefore in step 3 of the algorithm $\hat{A} = \hat{C}\hat{C}^T = A$.

□

B.1.3 Proof of Theorem 3

Proof. The proof is by construction. First, consider the case when A is full rank and $r = n$. In this case, we simply sample the full matrix and $A1 - A3$ trivially hold. Further, $|\Omega| = n^2 = nr < n(2r + 1)$.

Now, consider the case when $r < n$. Let Ω index the set of $k = n - r$ principal submatrices running down the diagonal of A, such that for $l \in \{1, \dots, k\}$, $\rho_l = \{1, \dots, r + 1\} + \{l - 1\}$.[2] We can verify that

$$\boldsymbol{\rho}\{\Omega\} = \boldsymbol{\rho}\{\cup_{l=1}^k \Omega_l\} = \cup_{l=1}^k \rho_l = [n],$$

meeting $A1$. Further, by construction each Ω_l satisfies $A2$, and if we let $\tau_1, \dots, \tau_k = 1 \dots, k$, we have that for $i \geq 2$

$$|\rho_{\tau_i} \cap \left(\cup_{j \in \{1,\dots,i-1\}}\rho_{\tau_j}\right)| = |\rho_{\tau_i} \cap \rho_{\tau_{i-1}}| = r,$$

[2]We will use the notation $\{\cdot\} + \{\cdot\}$ indicates set addition.

establishing that $A3$ holds as well. Having established that $A1 - A3$ hold for the proposed Ω, we count the number of indexed entries. The set Ω_1 indexes $(r+1)^2$ entries, and each further Ω_k for $k \geq 2$ indexes an additional $2r+1$ entries not index by earlier Ω_k. We can formally count these entries as

$$\begin{aligned}
|\Omega| &= |\Omega_1| + \sum_{j=2}^{n-r} \left| \Omega_j \setminus \left(\Omega_j \cap \left[\cup_{l=1}^{j-1} \Omega_l \right] \right) \right| \\
&= (r+1)^2 + \sum_{j=2}^{n-r} (2r+1) \\
&= (r+1)^2 + (n-r-1)(2r+1) \\
&= 2nr + n - r^2 - r \\
&\leq n(2r+1).
\end{aligned}$$

□

B.1.4 Proof of Theorem 4

Proof. For the rank r matrix $A \in \mathcal{S}_+^n$, fix $C \in \mathbb{R}^{n \times r}$ such that $A = CC^T$. By lemma 15, such a C must exist. Now, define $C_l = C(\rho_l, :)$. By assumption $A2$, each $A_l \in \mathcal{S}_+^{n_l}$, so $A_l = C_l C_l^T$ and by assumption $A4$, **rank**$\{A_l\} = r > 0$ for all l, as **rank**$\{A(\iota_l, \iota_l)\} = r$ for all $l \geq 2$.

Additionally, by assumption $A6$, $\tilde{A}_l \in \mathcal{S}_+^{n_l}$, by assumption $A5$, **rank**$\left\{\tilde{A}_l\right\} \geq r$, and $\|A_l - \tilde{A}_l\|_F \leq \epsilon < 1$ for all l. Further, we have $\min\{\min_{i \in [r-1],} |\lambda_{l,i} - \lambda_{l,i+1}|, \lambda_{l,r}\} \geq \delta$ for $l \in \{1, \ldots, k\}$, and we also have $\epsilon < \delta/2$. Then by lemma 29, if each $\hat{C}_l$ is formed according to steps 1 and 2a of algorithm[3] 1

$$\min_{W : WW^T = I} \|C_l - \hat{C}_l W\|_F \leq L\sqrt{r\epsilon},$$

for $L = \sqrt{1 + \frac{16\zeta}{\delta^2} + \frac{8\sqrt{2}\zeta^{1/2}}{\delta^{3/2}}}$, where we have used the assumption $\zeta \geq \lambda_{l,1}$ for all l.

Forming $\hat{C}$ from the $\hat{C}_l$ matrices in steps 1 and 2 of algorithm 1 is equivalent to forming $\hat{C}$ from the same $\hat{C}_l$ matrices with algorithm 3 in corollary 33 according to the ordering given by $\tau_1, \ldots, \tau_k$. Thus, we can use 33 to bound the error on $\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F$ at the end of step 2 of algorithm 1.

Corollary 33 requires that $\|C(\rho_l, :)\| \leq b$ for some $b \geq 0$ and $\min_{W:WW^T=I} \left\| C_l - \hat{C}_l W \right\|_F \leq \epsilon^*/b \leq 1$ for some $0 \leq \epsilon^* \leq b$ for all l.

By assumption $b \geq \max_{l \in [k]} \|C_l^*\|_F$ for some C_l^* such that $A_l = C_l^* C_l^{*T}$. However, we can show that there exists an orthogonal O such that $C_l = C_l^* O$ for any C_l and C_l^* such that $C_l^* C_l^{*T} = A_l = C_l C_l^T$. The Frobenius norm is a unitarily invariant norm, so if $b \geq \max_{l \in [k]} \|C_l^*\|_F$, then $b \geq \max_{l \in [k]} \|C_l\|_F$, as defined above, and it must be that $\|C(\rho_l, :)\|_F \leq b$ for all l. Defining $\epsilon^* = L\sqrt{r\epsilon}$, and noting that our theorem requires $\sqrt{r\epsilon} < b$, we can verify

$$\frac{\epsilon^*}{b} = \frac{L\sqrt{r\epsilon}}{b} \leq \frac{\sqrt{r\epsilon}}{b} < 1,$$

satisfying the requirements of the lemma.

Corollary 33 also requires that $C(\iota_l, :)$ and $\hat{C}_l(\phi_l, :)^T \hat{C}(\iota_l, :)$ each are of rank r for $l \in 2, \ldots, k$. However, this assumption is also met. To see this, note that by assumptions $A3$ and $A4$ $A(\iota_l, \iota_l) = C(\iota_l, :)C(\iota_l, :)^T$ is of rank r for all $l \geq 2$ so $C(\iota_l, :)$ must be of rank r for all $l \geq 2$. The requirement that $\hat{C}_l(\phi_l, :)^T \hat{C}(\iota_l, :)$ is of rank r for all $l \geq 2$

[3]Algorithm 1 never explicitly forms $\hat{C}_1$, but for clarity and brevity, we will define $\hat{C}(\rho_{\tau_1}, :)$ to be $\hat{C}(\rho_{\tau_1}, :) = \hat{C}_1$.

is satisfied by the assumptions of the theorem. Finally, corollary 33 also requires that $\cup_{l=1}^{k} \rho_l = [n]$, which is met by assumption $A1$.

Therefore, we can use corollary 33 to bound $\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F$ as

$$\begin{aligned}\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F &\leq [4 + 12/v]^{k-1} \epsilon^* \\ &= [4 + 12/v]^{k-1} L\sqrt{r\epsilon}\end{aligned}$$

when $v \leq \sigma_r^2(C(\iota_l, :))/b^2$ for all l. Note that $\lambda_r(A(\iota_l, \iota_l)) = \sigma_r^2(C(\iota_l, :))$, so this condition on v is equivalent to $v \leq \lambda_r(A(\iota_l, \iota_l))/b^2$ for all l.

To obtain a bound on $\left\| A - \hat{A} \right\|_F$, we apply lemma 25, to find

$$\begin{aligned}\left\| A - \hat{A} \right\|_F &= \left\| CC^T - \hat{C}\hat{C}^T \right\| \\ &\leq 2 \left\| C \right\|_F \left([4 + 12/v]^{k-1} L\sqrt{r\epsilon} \right) + \left([4 + 12/v]^{k-1} L\sqrt{r\epsilon} \right)^2 \\ &= 2 \left\| C \right\|_F \left([4 + 12/v]^{k-1} L\sqrt{r\epsilon} \right) + [4 + 12/v]^{2k-2} L^2 r\epsilon,\end{aligned}$$

defining $G = 4 + 12/v$, we have

$$\left\| A - \hat{A} \right\|_F \leq 2G^{k-1} L \left\| C \right\|_F \sqrt{r\epsilon} + G^{2k-2} L^2 r\epsilon. \tag{B.1}$$

Note that the statement of the main theorem in the body of the thesis allows C to be any arbitrary n by r matrix such that $A = CC^T$. We note that indeed any such C will do as if $A = CC^T = C'C'^T$ for some $C' \in \mathbb{R}^{n \times r}$, there is an orthogonal transformation relating C and C'. The Frobenius norm is a unitarily invariant norm, so $\|C\|_F = \|C'\|_F$, which completes the proof.

□

B.2 Technical Results

Lemma 15. *For any $A \in \mathcal{S}_+^n$ such that* **rank**$\{A\} \leq r$*, there exists a matrix $C \in \mathbb{R}^{n \times r}$ such that $A = CC^T$.*

Proof. First, assume **rank**$\{A\} = 0$. Then if C is a appropriately sized zero vector, the lemma is satisfied. Now, assume **rank**$\{A\} > 0$. By the spectral theorem for real symmetric matrices, any rank $p > 0$ symmetric, real matrix can be decomposed as

$$A = E\Lambda E^T,$$

for some diagonal $\Lambda \in \mathbb{R}^{p \times p}$ containing the p non-zero eigenvalues of A and $E \in \mathbb{R}^{n \times p}$, containing the corresponding eigenvectors. Further, the p non-zero eigenvalues of A must all be greater than 0 as A is PSD. Define $C = E\Sigma^{1/2}$. By construction $C \in \mathbb{R}^{n \times p}$ and we have $CC^T = E\Sigma^{1/2}(E\Sigma^{1/2})^T = E\Sigma E^T = A$. If $p = r$, this establishes the existence of a $C \in \mathbb{R}^{n \times r}$ such that $A = CC^T$. If $p < r$, a $C \in \mathbb{R}^{n \times r}$ such that $A = CC^T$ can be constructed by adding $r - p$ columns of zeros to the matrix $E\Sigma^{1/2}$. □

Lemma 16. *Let $C \in \mathbb{R}^{n \times p}$ be a rank $r > 0$ matrix. For the orthogonal matrix $W \in \mathbb{R}^{p \times p}$, consider the equality $C = CW$. If $r < p$, then W is non-unique. Further, $W = I$ uniquely if and only if $r = p$.*

Proof. We begin with the if and only if statement. Assume $r = p$. If $r = p$, it must be that $n \geq p$. We write the singular value decomposition of C as $C = U\Sigma V^T$ for some $U \in \mathbb{R}^{n\times p}$ with orthonormal columns, orthogonal $V \in \mathbb{R}^{p\times p}$ and diagonal $\Sigma \in \mathbb{R}^{p\times p}$ with the p non-zero singular values of C along its diagonal. We now solve for W in the equation $C = CW$ to show that uniquely $W = I$. Specifically, we have

$$\begin{aligned} CW &= C \\ U\Sigma V^T W &= U\Sigma V^T \\ (V\Sigma^{-1}U^T)U\Sigma V^T W &= (V\Sigma^{-1}U^T)U\Sigma V^T \\ W &= I. \end{aligned}$$

We now consider $r < p$. Denote the singular value decomposition of C as $C = U\Sigma V^T$. We now need to consider the possibility that $n < p$, so define $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{p\times p}$ to be orthogonal matrices and let $\Sigma \in \mathbb{R}^{n\times p}$ be a matrix with the singular values of C in the appropriate locations. Specifically, the diagonal of Σ will contain the r singular values of C and $p - r$ zero values along its diagonal. We again consider solutions to the equation $U\Sigma V^T W = U\Sigma V^T$. We can left multiply by U^T to obtain $\Sigma V^T W = \Sigma V^T$. Let $W = VO$ for some orthogonal $O \in \mathbb{R}^{p\times p}$. Let $O(i,:) = V(:,i)^T$ if $\Sigma(i,i) \neq 0$. Then it can be verified that $\Sigma V^T = \Sigma V^T W$, regardless of the values of the remaining rows of O. Furter, the remaining rows of O can be chosen arbitrarily as long as O remains orthogonal. In the case of a single missing row, the missing row is only determined up to multiplication by 1 or -1. In the case of more than one missing row, there is greater flexibility. When O is non-unique, the product $W = VO$ will be non-unique, completing the proof.

□

Corollary 17. *Let $C_1, C_2 \in \mathbb{R}^{n\times r}$ be matrices such that $C_1 = C_2 W$ for some orthogonal $W \in \mathbb{R}^{r\times r}$. If the rank of C_1 is $r > 0$, then W is unique.*

Proof. The proof is by contradiction. First, assume there are two distinct orthogonal matrices W_1 and W_2, such that $C_1 = C_2 W_1$ and $C_1 = C_2 W_2$. Then it must be that

$$C_2 = C_2 W_1 W_2^T. \tag{B.2}$$

Next, note that if W_1 and W_2 are two distinct orthornormal matrices, $W_1 W_2^T \neq I$. However, by assumption C_2 is of rank $r > 0$ and by lemma 16 (B.2) holds if and only if $W_1 W_2^T = I$, and we have achieved a contradiction.

□

Lemma 18. *For the rank $r > 0$ matrix $A \in \mathcal{S}_+^n$ define $C_1, C_2 \in \mathbb{R}^{n\times r}$ such that $A = C_1 C_1^T$ and $A = C_2 C_2^T$. Then there exists a unique orthogonal $W \in \mathbb{R}^{r\times r}$ such that $C_1 = C_2 W$.*

Proof. First note that by assumption $\textbf{rank}\{A\} = r$, so C_1 and C_2 must also be of rank r. Further, it must be that $\textbf{rank}\{A\} \leq n$, so $r \leq n$. Denote the singular value decomposition of C_1 as $C_1 = U_1\Sigma_1 V_1^T$, for $U_1 \in \mathbb{R}^{n\times r}$ with orthonormal columns, diagonal $r \times r$ matrix Σ_1, with the non-zero singular values of C_1 along its diagonal, and orthogonal $V_1 \in \mathbb{R}^{r\times r}$. Use a similar notation for the SVD of C_2. With these preliminaries out of the way, we will first establish that W exists and is orthogonal and then call on corollary 17 to establish its uniqueness.

First, we establish that W exists by noting that we can solve for it. All of the singular values of C_1 and C_2 must be strictly greater than 0, as C_1 and C_2 are of rank $r > 0$. Therefore, Σ_1^{-1} and Σ_2^{-1} are well defined. Thus, we can solve for W as $W = V_2\Sigma_2^{-1}U_2^T U_1\Sigma_1 V_1^T$.

To establish that W must be orthogonal let W be a matrix such that $C_1 = C_2 W$ and note that we have

$$A = C_1 C_1^T = C_2 W W^T C_2^T = C_2 C_2^T = A,$$

so clearly $C_2 C_2^T = C_2 W W^T C_2^T$.

Using our notation for the SVD of $C2$, we can then write

$$U_2\Sigma_2 V_2^T I V_2\Sigma_2 U_2^T = U_2\Sigma_2 V_2^T W W^T V_2\Sigma_2 U_2^T. \tag{B.3}$$

Noting again that Σ_2^{-1} is well defined, we can then left multiple both sides of (B.3) by $V_2\Sigma_2^{-1}U_2^T$ and right multiply by $U_2\Sigma_2^{-1}V_2^T$ to find,

$$I = WW^T,$$

establishing that W must be orthogonal. Having established that there exists an orthogonal W such that $C_1 = C_2W$, that C_1 and C_2 are of rank $r > 0$, the uniqueness of W follows directly from corollary 17. □

Lemma 19. *Consider the rank $r > 0$ matrix $A \in \mathcal{S}_+^n$. Define the index sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $A(\mathcal{I}_1, \mathcal{I}_1)$ and $A(\mathcal{I}_2, \mathcal{I}_2)$ are submatrices of A and $A(\mathcal{I}_1 \cup \mathcal{I}_2, \mathcal{I}_1 \cup \mathcal{I}_2) = A$. Let $\iota = \mathcal{I}_1 \cap \mathcal{I}_2$. Let $n_1 = |\mathcal{I}_1|$, and $n_2 = |\mathcal{I}_2|$.*

Define $C_1 \in \mathbb{R}^{n_1\times r}$ and $C_2 \in \mathbb{R}^{n_2\times r}$ such that $A(\mathcal{I}_1, \mathcal{I}_1) = C_1C_1^T$ and $A(\mathcal{I}_2, \mathcal{I}_2) = C_2C_2^T$.

Further, let ϕ_1 and ϕ_2 index the rows of C_1 and C_2, respectively, such that $C_1(\phi_1, :)C_1(\phi_1, :)^T = A(\iota, \iota) = C_2(\phi_2, :)C_2(\phi_2, :)^T$. Finally, let η_2 index the rows of C_2 such that $A(\mathcal{I}_2 \setminus \iota, \mathcal{I}_2 \setminus \iota) = C_2(\eta_2, :)C_2(\eta_2, :)^T$. Then if the rank of $A(\iota, \iota)$ is equal to r, there exists a unique orthogonal

$$W : C_1(\phi_1, :) = C_2(\phi_2, :)W \tag{B.4}$$

such that $A = C'C'^T$ when $C' \in \mathbb{R}^{n\times r}$ is formed as

$$\begin{aligned} C'(\mathcal{I}_1, :) &= C_1 \\ C'(\mathcal{I}_2 \setminus \iota, :) &= C_2(\eta_2, :)W. \end{aligned}$$

Proof. Note that

$$C_1(\phi_1, :)C_1(\phi_1, :)^T = C_2(\phi_2, :)C_2(\phi_2, :)^T = A(\iota, \iota).$$

Further, by assumption $A(\iota, \iota)$ is of rank $r > 0$, so $C_1(\phi_1, :)$ and $C_2(\phi_2, :)$ must be as well. Then the existence, ortogonality and uniqueness of W follow from lemma 18.

Next, we establish that $C'C'^T = A$. First note that

$$C'(\mathcal{I}_1, :)C'(\mathcal{I}_1, :)^T = C_1C_1^T = A(\mathcal{I}_1, \mathcal{I}_1),$$

and

$$\begin{aligned} C'(\mathcal{I}_2 \setminus \iota, :)C'(\mathcal{I}_2 \setminus \iota, :)^T &= C_2(\eta_2, :)WW^TC_2(\eta_2, :)^T \\ &= C_2(\eta_2, :)C_2(\eta_2, :)^T \\ &= A(\mathcal{I}_2 \setminus \iota, \mathcal{I}_2 \setminus \iota). \end{aligned}$$

All that remains is to show that $C'(\mathcal{I}_2 \setminus \iota, :)C'(\mathcal{I}_1, :)^T = A(\mathcal{I}_2 \setminus \iota, \mathcal{I}_1)$. We do this by reference to an arbitrary matrix $D \in \mathbb{R}^{n\times r}$ such that $A = DD^T$. It must be that $A(\mathcal{I}_1, \mathcal{I}_1) = D(\mathcal{I}_1, :)D(\mathcal{I}_1, :)^T$ and $A(\mathcal{I}_2, \mathcal{I}_2) = D(\mathcal{I}_2, :)D(\mathcal{I}_2, :)^T$. Additionally, we also know that $A(\mathcal{I}_1, \mathcal{I}_1) = C_1C_1^T$ and $A(\mathcal{I}_2, \mathcal{I}_2) = C_2C_2^T$.

Further, the rank r matrix $A(\iota,\iota)$ is a submatrix of $A(\mathcal{I}_1,\mathcal{I}_1)$ and $A(\mathcal{I}_2,\mathcal{I}_2)$ so $A(\mathcal{I}_1,\mathcal{I}_1)$ and $A(\mathcal{I}_2,\mathcal{I}_2)$ must also be of rank r, from which it follows that $D(\mathcal{I}_1,:)$, $D(\mathcal{I}_2,:)$, C_1 and C_2 are also of rank r.

From lemma 18 it then follows that there exist unique orthogonal O_1 and O_2 such that $D(\mathcal{I}_1,:) = C_1O_1$ and $D(\mathcal{I}_2,:) = C_2O_2$. Further, we have

$$\begin{aligned} C_1(\phi_1,:)C_1(\phi_1,:)^T &= C_2(\phi_2,:)WC_1(\phi_1,:)^T \\ &= D(\iota,:)O_2^TWO_1D(\iota,:)^T \\ &= A(\iota,\iota), \end{aligned}$$

which establishes that $D(\iota,:)O_2^TWO_1D(\iota,:)^T = A(\iota,\iota)$. Further, $D(\iota,:)$ is a rank $r > 0$ matrix of size $|\iota|$ by r. Therefore, for the matrix M it must be that $D(\iota,:)MD(\iota,:)^T = A(\iota,\iota) = D(\iota,:)D(\iota,:)^T$ if and only if $M = I$. Therefore, $D(\iota,:)O_2^TWO_1D(\iota,:)^T = A(\iota,\iota)$ if and only if $O_2WO_1^T = I$.

Now, consider, $C'(\mathcal{I}_2\setminus\iota,:)C'(\mathcal{I}_1,:)^T$. We can write

$$\begin{aligned} C'(\mathcal{I}_2\setminus\iota,:)C'(\mathcal{I}_1,:)^T &= C_2(\eta_2,:)WC_1^T \\ &= D(\mathcal{I}_2\setminus\iota,:)O_2^TWO_1D(\mathcal{I}_1,:)^T \\ &= D(\mathcal{I}_2\setminus\iota,:)ID(\mathcal{I}_1,:)^T \\ &= A(\mathcal{I}_2\setminus\iota,\mathcal{I}_1). \end{aligned}$$

Having shown that $A(\mathcal{I}_1,\mathcal{I}_1) = C'(\mathcal{I}_1,:)C'(\mathcal{I}_1,:)^T$, $A(\mathcal{I}_2\setminus\iota,\mathcal{I}_2\setminus\iota) = C'(\mathcal{I}_2\setminus\iota,:)C'(\mathcal{I}_2\setminus\iota,:)^T$ and $A(\mathcal{I}_2\setminus\iota,\mathcal{I}_1) = C'(\mathcal{I}_2\setminus\iota,:)C'(\mathcal{I}_1,:)^T$, we can conclude that $A(\mathcal{I}_1\cup\mathcal{I}_2,\mathcal{I}_1\cup\mathcal{I}_2) = C'C'^T$. Further, $A(\mathcal{I}_1\cup\mathcal{I}_2,\mathcal{I}_1\cup\mathcal{I}_2)$ must be the entirety of A as by assumption $A(\mathcal{I}_1\cup\mathcal{I}_2,\mathcal{I}_1\cup\mathcal{I}_2) = A$, which completes the proof.

□

Lemma 20. *Let $A \in \mathcal{S}_+^n$ be a rank $r > 0$ matrix. Define the index sets $\mathcal{I}_1 \subset [n]$ and $\mathcal{I}_2 \subset [n]$ and let $\iota = \mathcal{I}_1 \cap \mathcal{I}_2$. Let Ω denote the set of observed entries of A and assume $\Omega = \mathcal{I}_1\times\mathcal{I}_1 \cup \mathcal{I}_2\times\mathcal{I}_2$ and $\Omega \neq [n]\times[n]$. Assume* ***rank***$\{A(\mathcal{I}_1,\mathcal{I}_1)\}$ = ***rank***$\{A(\mathcal{I}_2\setminus\iota,\mathcal{I}_2\setminus\iota)\} = r$. *Then it is necessary that $\mathcal{I}_1\cup\mathcal{I}_2 = [n]$ and* ***rank***$\{A(\iota,\iota)\} = r$, *where it is understood that* ***rank***$\{A(\iota,\iota)\} = 0$ *if $\iota = \emptyset$, for there to be a unique rank r completion of A.*

Proof. First, if $\mathcal{I}_1\cup\mathcal{I}_2 \neq [n]$, the set Ω fails to index a complete row and column of A, which we compactly express as $\{(i,j),(j,i) : j \in [n]\}\cap\Omega = \emptyset$ for some $i \in [n]$. The matrix A is SPSD, and by lemma 15 we can decompose A as $A = CC^T$ for some $C \in \mathbb{R}^{n\times r}$. For any such C, we can write the individual entries of A as $A(i,j) = C(i,:)C(j,:)^T$ for all $i,j \in [n]$. For row i of any such C, the set of entries $\{A_{ij}, A_{ji} : j \in [n]\}$ then provide the only set of equality constraints for $C(i,:)$. Now if $\{(i,j),(j,i) : j \in [n]\}\cap\Omega = \emptyset$ for some $i \in [n]$, this implies there are no equality constraints to learn $C(i,;)$. Therefore, there are an infinite number of C matrices such that $A_{ij} = (CC^T)_{ij}, \forall (i,j) \in \Omega$ but $A \neq CC^T$. We now show that **rank**$\left\{\hat{A}\right\} = r$. First, $C \in \mathbb{R}^{n\times r}$, so it must be **rank**$\left\{CC^T\right\} \leq r$. Additionally, $\mathcal{I}_1\times\mathcal{I}_1 \subseteq \Omega$, so it must be that $\hat{A}(\mathcal{I}_1,\mathcal{I}_1) = A(\mathcal{I}_1,\mathcal{I}_1)$. By assumption **rank**$\{A(\mathcal{I}_1,\mathcal{I}_1)\} = r$, and therefore, **rank**$\left\{\hat{A}\right\} \geq r$. We then have $r \leq$ **rank**$\left\{\hat{A}\right\} \leq r$ and conclude **rank**$\left\{\hat{A}\right\} = r$.

The rest of this proof now considers the case when $\mathcal{I}_1\cup\mathcal{I}_2 = [n]$ but **rank**$\{A(\iota,\iota)\} < r$. The matrix A is of rank r, and $A(\iota,\iota)$ is a submatrix of A. Therefore, it must be that **rank**$\{A(\iota,\iota)\} \leq r$. We will show that under the assumptions of the lemma when **rank**$\{A(\iota,\iota)\} < r$ we can find two or more rank r completions for A which match the entries of A in Ω but differ on their entries outside of Ω, establishing that there is no unique rank r completion for A from the set of observed entries in Ω.

The matrix A is PSD, and we by lemma 15 we can decompose A as $A = CC^T$ for some $C \in \mathbb{R}^{n\times r}$.

We construct a matrix, $\hat{C} \in \mathbb{R}^{n\times r}$, as

$$\hat{C}(\mathcal{I}_1, :) = C(\mathcal{I}_1, :),$$
$$\hat{C}(\mathcal{I}_2 \setminus \iota, :) = C(\mathcal{I}_2 \setminus \iota, :)W,$$

for some orthogonal $W \in \mathbb{R}^{r \times r}$ such that

$$C(\iota, :) = C(\iota, :)W. \tag{B.5}$$

If $\iota = \emptyset$, then any orthogonal matrix will satisfy this constraint. If $\iota \neq \emptyset$ and rank **rank**$\{A(\iota, \iota)\} < r$, then such an orthogonal W still exists and is non-unique. To see this note that if the rank of $A(\iota, \iota)$ is less than r, the rank of $C(\iota, :)$ must also be less than r as $C(\iota, :)C(\iota, :)^T = A(\iota, \iota)$. By lemma 16 we can then conclude that there exists a non-unique orthogonal W such that $C(\iota, :) = C(\iota, :)W$.

Let $\hat{A} = \hat{C}\hat{C}^T$. The matrix $\hat{A}$ must agree with A on all entries indexed by Ω. To see this note

$$\begin{aligned}\hat{A}(\mathcal{I}_1, \mathcal{I}_1) &= \hat{C}(\mathcal{I}_1, :)\hat{C}(\mathcal{I}_1, :)^T \\ &= C(\mathcal{I}_1, :)C(\mathcal{I}_1, :)^T \\ &= A(\mathcal{I}_1, \mathcal{I}_1),\end{aligned}$$

and

$$\begin{aligned}\hat{A}(\mathcal{I}_2, \mathcal{I}_2) &= \hat{C}(\mathcal{I}_2, :)\hat{C}(\mathcal{I}_2, :)^T \\ &= C(\mathcal{I}_2, :)WW^T C(\mathcal{I}_2, :)^T \\ &= C(\mathcal{I}_2, :)C(\mathcal{I}_2, :)^T \\ &= A(\mathcal{I}_2, \mathcal{I}_2),\end{aligned}$$

where we have used the relations $\hat{C}(\mathcal{I}_2 \cap \iota, :) = C(\iota, :) = C(\iota, :)W$ and $\hat{C}(\mathcal{I}_2 \setminus \iota, :) = C(\mathcal{I}_2 \setminus \iota, :)W$ to conclude $\hat{C}(\mathcal{I}_2, :) = C(\mathcal{I}_2, :)W$. Note that $\Omega = \mathcal{I}_1 \times \mathcal{I}_1 \cup \mathcal{I}_2 \times \mathcal{I}_2$, so the above establishes that any $\hat{A}$ constructed in this way will be indeed be identical to A on all entries in Ω.

Now, we consider the portion of the matrix not indexed by Ω, that is $A(\mathcal{I}_1, \mathcal{I}_2 \setminus \iota)$. Note that as $\Omega \neq [n] \times [n]$, $A(\mathcal{I}_1, \mathcal{I}_2 \setminus \iota)$ consists of at least one entry of the matrix. Under the assumptions of the lemma, if W is non-unique then the product

$$\hat{A}(\mathcal{I}_1, \mathcal{I}_2 \setminus \iota) = \hat{C}(\mathcal{I}_1, :)\hat{C}(\mathcal{I}_2 \setminus \iota, :)^T = C(\mathcal{I}_1, :)W^T C(\mathcal{I}_2 \setminus \iota, :)^T$$

is non-unique. To establish this first note that $A(\mathcal{I}_1, \mathcal{I}_1)$ and $A(\mathcal{I}_2 \setminus \iota, \mathcal{I}_2 \setminus \iota)$ are by assumption rank r, so $C(\mathcal{I}_1, :)$ and $C(\mathcal{I}_2 \setminus \iota, :)$ must be rank r as well. Let W_1 and W_2 be two orthogonal matrices which satisfy (B.5). Then $C(\mathcal{I}_1, :)(W_1 - W_2)^T C(\mathcal{I}_2 \setminus \iota, :)^T$ is non-zero. To see this note that if $W_1 \neq W_2$, then there exists some $x \in \mathbb{R}^r$ such that $(W_1 - W_2)^T x \neq 0$. However, $C(\mathcal{I}_2 \setminus \iota, :)$ is rank r, so the columns of $C(\mathcal{I}_2 \setminus \iota, :)^T$ must span $\mathbb{R}^r$, and therefore it must be that $(W_1 - W_2)^T C(\mathcal{I}_2 \setminus \iota, :)^T \neq 0$. Finally, $C(\mathcal{I}_1, :) \in \mathbb{R}^{|n_1| \times r}$ is rank r so for any $y \in \mathbb{R}^r$, $C(\mathcal{I}_1, :)y = 0$ if and only if $y = 0$. Therefore, as $(W_1 - W_2)^T C(\mathcal{I}_2 \setminus \iota, :)^T$ is a non-zero matrix, it must be that $C(\mathcal{I}_1, :)(W_1 - W_2)^T C(\mathcal{I}_2 \setminus \iota, :)^T \neq 0$.

Thus, we have shown that when **rank**$\{A(\iota, \iota)\} < r$, we can construct two or more $\hat{A}$ matrices which are identical to A for all entries indexed by Ω but differ on entries not indexed by Ω. All that remains to show is that any such $\hat{A}$ is rank r. This is accomplished by noting that for any $\hat{A}$ formed from a $\hat{C}$ as described above, $\hat{A}(\mathcal{I}_1, \mathcal{I}_1) = A(\mathcal{I}_1, \mathcal{I}_1)$, and **rank**$\{A(\mathcal{I}_1, \mathcal{I}_1)\} = r$, so the rank of $\hat{A}$ must be at least r. Further, $\hat{A} = \hat{C}\hat{C}^T$, and $\hat{C} \in \mathbb{R}^{n \times r}$, so **rank**$\left\{\hat{C}\right\} \leq r$ and therefore **rank**$\left\{\hat{A}\right\} \leq r$. Thus $r \leq$ **rank**$\left\{\hat{A}\right\} \leq r$, and we conclude **rank**$\left\{\hat{A}\right\} = r$.

□

Corollary 21. *Let $\Omega \neq [n] \times [n]$ index the set of observed entries of a rank $r > 0$ matrix $A \in \mathcal{S}_{+}^{n}$. Suppose there exists a $\rho_1, \dots, \rho_k$ for $k \geq 2$ such that $\cup_{i=1}^{k} \rho_i \times \rho_i = \Omega$, $\rho_i \not\subseteq \rho_j$ for all pairs $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq j\}$ and $\rho_i \cap \rho_j = \emptyset$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | \ |i-j| > 1\}$.*

*Define $\iota_i = \rho_i \cap \rho_{i-1}$ for $i \in \{2, \dots, k\}$. Assume **rank**$\{A(\rho_1, \rho_1)\} = r$ and **rank**$\{A(\rho_i \setminus \iota_i, \rho_i \setminus \iota_i)\} = r$ for all $i \in \{2, \dots, k\}$.*

*Then it is necessary that $\cup_{i=1}^{k} \rho_i = [n]$ and **rank**$\{A(\iota_i, \iota_i)\} = r$ for all $i \in \{2, \dots, k\}$, where it is understood that **rank**$\{A(\iota_i, \iota_i)\} = 0$ if $\iota_i = \emptyset$, for there to be a unique rank r completion of A.*

Proof. Fix $i^* \geq 2$. Define $\rho_{i-1}^* = \cup_{j=1}^{i^*-1} \rho_j$ and $\rho_i^* = \cup_{j=i^*}^{k} \rho_j$ and $\Omega^* = \rho_{i-1}^* \times \rho_{i-1}^* \cup \rho_i^* \times \rho_i^*$.

By construction $\rho_{i-1}^* \in [n]$ and $\rho_i^* \in [n]$. Further, we prove that $\Omega^* \neq [n] \times [n]$ by contradiction. Assume that $\Omega^* = [n] \times [n]$. Then this would require that either $\rho_{i-1}^* = [n]$ or $\rho_i^* = [n]$, which would imply $\rho_{i-1}^* \subseteq \rho_i^*$ or $\rho_{i-1}^* \supseteq \rho_i^*$. However, since we assume $\rho_i \cap \rho_j = \emptyset$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | \ |i-j| > 1\}$, this would require $\rho_{i^*-1} \subseteq \rho_{i^*}$ or $\rho_{i^*-1} \supseteq \rho_{i^*}$ both of which contradict the assumption that $\rho_i \not\subseteq \rho_j$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq j\}$.

Thus, we have two sets, ρ_{i-1}^* and ρ_i^*, which index A as required by lemma 20. Now assume that $\cup_{i=1}^{k} \rho_i \neq [n]$. Then $\rho_{i-1}^* \cup \rho_i^* \neq [n]$, so by lemma 20, A cannot be completed from the entries indexed by Ω^*.

Now, assume that $\cup_{i=1}^{k} \rho_i = [n]$ so $\rho_{i-1}^* \cup \rho_i^* = [n]$, but there exists an $i^* \geq 2$ such that **rank**$\{A(\iota_{i^*}, \iota_{i^*})\} < r$. Define $\iota^* = \rho_{i-1}^* \cap \rho_i^*$.

Then

$$\begin{aligned}
\textbf{rank}\{A(\iota^*, \iota^*)\} &= \textbf{rank}\big\{A(\rho_{i-1}^* \cap \rho_i^*, \rho_{i-1}^* \cap \rho_i^*)\big\} \\
&= \textbf{rank}\{A(\rho_{i^*-1} \cap \rho_{i^*}, \rho_{i^*-1} \cap \rho_{i^*})\} \\
&= \textbf{rank}\{A(\iota_{i^*}, \iota_{i^*})\} \\
&< r.
\end{aligned}$$

Further, $\iota^* \geq 2$, so $\rho_1 \subseteq \rho_{i-1}^*$. By assumption the rank of $A(\rho_1, \rho_1)$, which is a submatrix of $A(\rho_{i-1}^*, \rho_{i-1}^*)$, is r, and so it must be that **rank**$\big\{A(\rho_{i-1}^*, \rho_{i-1}^*)\big\} = r$. Additionally, $\rho_{i^*} \subseteq \rho_i^*$ and $\iota^* = \iota_{i^*}$, so $\rho_{i^*} \setminus \iota_{i^*} \subseteq \rho_i^* \setminus \iota^*$. This establishes that $A(\rho_{i^*} \setminus \iota_{i^*}, \rho_{i^*} \setminus \iota_{i^*})$ is a submatrix of $A(\rho_i^* \setminus \iota^*, \rho_i^* \setminus \iota^*)$. By assumption **rank**$\{A(\rho_{i^*} \setminus \iota_{i^*}, \rho_{i^*} \setminus \iota_{i^*})\} = r$, so it must be that **rank**$\{A(\rho_i^* \setminus \iota^*, \rho_i^* \setminus \iota^*)\} = r$. Therefore, by lemma 20 there is again no unique rank r completion of A from the entries indexed by Ω^*.

Finally,

$$\begin{aligned}
\Omega &= \cup_{j=1}^{k} \rho_j \times \rho_j \\
&= \left(\cup_{j=1}^{i^*-1} \rho_j \times \rho_j\right) \cup \left(\cup_{j=i^*}^{n} \rho_j \times \rho_j\right) \\
&\subseteq \rho_{i-1}^* \times \rho_{i-1}^* \cup \rho_i^* \times \rho_i^* \\
&= \Omega^*.
\end{aligned}$$

Therefore, the set Ω^* is a superset of Ω. If no unique rank r completion can be found from a superset of Ω, then the entries indexed by Ω must themselves contain insufficient information to allow for a unique rank r completion of A, completing the proof.

□

Corollary 22. *Let $\Omega \neq [n] \times [n]$ index the set of observed entries of a rank $r > 0$ matrix $A \in \mathcal{S}_{+}^{n}$. Suppose there exists a $\rho_1, \dots, \rho_k$ for $k \geq 2$ such that $\cup_{i=1}^{k} \rho_i \times \rho_i = \Omega$ and $\rho_i \not\subseteq \rho_j$ for all pairs $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq j\}$. Assume $(\rho_i \setminus \rho_1) \cap (\rho_j \setminus \rho_1) = \emptyset$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq 1, j \neq 1, i \neq j\}$.*

*Define $\iota_i = \rho_i \cap \rho_1$ for all $i \in \{2, \dots, k\}$. Assume **rank**$\{A(\rho_1, \rho_1)\} = r$ and **rank**$\{A(\rho_i \setminus \iota_i, \rho_i \setminus \iota_i)\} = r$ for all $i \in \{2, \dots, k\}$.*

*Then it is necessary that $\cup_{i=1}^{k} \rho_i = [n]$ and **rank**$\{A(\iota_i, \iota_i)\} = r$ for all $i \in \{2, \dots, k\}$, where it is understood that **rank**$\{A(\iota_i, \iota_i)\} = 0$ if $\iota_i = \emptyset$, for there to be a unique rank r completion of A.*

Proof. Fix $i^* \geq 2$. Define $\rho_{\setminus i^*} = \cup_{j \neq i^*} \rho_j$ and $\Omega^* = \rho_{\setminus i^*} \times \rho_{\setminus i^*} \cup \rho_{i^*} \times \rho_{i^*}$.

By construction $\rho_{\setminus i^*} \in [n]$ and $\rho_{i^*} \in [n]$. Further, we prove that $\Omega^* \neq [n] \times [n]$ by contradiction. Assume that $\Omega^* = [n] \times [n]$. Then this would require that either $\rho_{\setminus i^*} = [n]$ or $\rho_{i^*} = [n]$, which would imply $\rho_{\setminus i^*} \supseteq \rho_{i^*}$ or $\rho_{\setminus i^*} \subseteq \rho_{i^*}$. However, since we assume $(\rho_i \setminus \rho_1) \cap (\rho_j \setminus \rho_1) = \emptyset$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq 1, j \neq 1, i \neq j\}$, this would imply that $\rho_1 \supseteq \rho_{i^*}$ for some $i \geq 2$ or $\rho_1 \subseteq \rho_{i^*}$ for some $i \geq 2$, both of which contradict the assumption that $\rho_i \not\subseteq \rho_j$ for any pair $(i,j) \in \{(i,j) \in [k] \times [k] | i \neq j\}$.

Thus, we have two sets, $\rho_{\setminus i^*}$ and ρ_{i^*}, which index A as required by lemma 20. Now assume that $\cup_{i=1}^k \rho_i \neq [n]$. Then $\rho_{\setminus i^*} \cup \rho_{i^*} \neq [n]$, so by lemma 20, A cannot be completed from the entries indexed by Ω^*.

Now, assume that $\cup_{i=1}^k \rho_i = [n]$ so $\rho_{\setminus i^*} \cup \rho_{i^*} = [n]$, but there exists an i^* such that $\textbf{rank}\{A(\iota_{i^*}, \iota_{i^*})\} < r$. Define $\iota^* = \rho_{\setminus i^*} \cap \rho_{i^*}$.

Then

$$\begin{aligned}
\textbf{rank}\{A(\iota^*, \iota^*)\} &= \textbf{rank}\big\{A(\rho_{\setminus i^*} \cap \rho_{i^*}, \rho_{\setminus i^*} \cap \rho_{i^*})\big\} \\
&= \textbf{rank}\{A(\rho_1 \cap \rho_{i^*}, \rho_1 \cap \rho_{i^*})\} \\
&= \textbf{rank}\{A(\iota_{i^*}, \iota_{i^*})\} \\
&< r.
\end{aligned}$$

Further, $\rho_1 \subseteq \rho_{\setminus i^*}$, so $A(\rho_1, \rho_1)$, which is rank r by assumption, is a submatrix of $A(\rho_{\setminus i^*}, \rho_{\setminus i^*})$, which must then also be rank r. Additionally, $\iota^* = \iota_{i^*}$, so $\rho_{i^*} \setminus \iota^* = \rho_{i^*} \setminus \iota_{i^*}$. By assumption $A(\rho_{i^*} \setminus \iota_{i^*}, \rho_{i^*} \setminus \iota_{i^*})$ is rank r, so $A(\rho_{i^*} \setminus \iota^*, \rho_{i^*} \setminus \iota^*)$ is also rank r.

Therefore, by lemma 20 there is again no unique rank r completion of A from the entries indexed by Ω^*. However,

$$\begin{aligned}
\Omega &= \cup_{j=1}^k \rho_j \times \rho_j \\
&= (\cup_{j \neq i^*} \rho_j \times \rho_j) \cup (\rho_{i^*} \times \rho_{i^*}) \\
&\subseteq \rho_{\setminus i^*} \times \rho_{\setminus i^*} \ \cup \rho_i^* \times \rho_i^* \\
&= \Omega^*.
\end{aligned}$$

The set Ω^* is a superset of Ω. If no unique rank r completion can be found from a superset of Ω, then the entries indexed by Ω must themselves contain insufficient information to allow for a unique rank r completion of A, completing the proof.

□

Corollary 23. *Let $\Omega \neq [n] \times [n]$ index the set of observed entries of a rank $r > 0$ matrix $A \in \mathcal{S}_+^n$. Assume there exists an $\mathcal{I} \subseteq [n]$ and $\rho \subseteq [n]$ such that $\rho \not\subseteq \mathcal{I}$, $\mathcal{I} \not\subseteq \rho$, $\rho \times \rho \subseteq \Omega$ and $\Omega \subseteq \rho \times \rho \cup \mathcal{I} \times \mathcal{I}$. Define $\iota = \rho \cap \mathcal{I}$. Assume **rank**$\{A(\rho, \rho)\} = r$ and there exists at least one observed rank r submatrix in $A(\mathcal{I} \setminus \iota, \mathcal{I} \setminus \iota)$.*

*Then it is necessary that $\rho \cup \mathcal{I} = [n]$ and **rank**$\{A(\iota, \iota)\} = r$, where it is understood that **rank**$\{A(\iota, \iota)\} = 0$ if $\iota = \emptyset$, for there to be a unique rank r completion of A.*

Proof. The proof is a straightforward application of lemma 20.

Define $\Omega^* = \rho \times \rho \cup \mathcal{I} \times \mathcal{I}$. Note that $\Omega^* \neq [n] \times [n]$, as by assumption $\mathcal{I} \subseteq [n]$, $\rho \subseteq [n]$, $\rho \not\subseteq \mathcal{I}$ and $\mathcal{I} \not\subseteq \rho$. Assume $\rho \cup \mathcal{I} \neq [n]$. Then by lemma 20, A cannot be recovered from the entries indexed by Ω^*.

Now assume that $\rho \cup \mathcal{I} = [n]$, but $\textbf{rank}\{A(\iota, \iota)\} < r$. By assumption, $\textbf{rank}\{A(\rho, \rho)\} = r$ and there exists at least one observed rank r submatrix in $A(\mathcal{I} \setminus \iota, \mathcal{I} \setminus \iota)$, so it must be that $\textbf{rank}\{A(\mathcal{I} \setminus \iota, \mathcal{I} \setminus \iota)\} = r$. Then by lemma 20, A cannot be recovered from the entries indexed by Ω^*.

However, $\Omega^* \supseteq \Omega$. If no unique rank r completion can be found from a superset of Ω, then the entries indexed by Ω must themselves contain insufficient information to allow for a unique rank r completion of A, completing the proof.

□

Lemma 24. *For any matrix pair $C \in \mathbb{R}^{n\times p}$ and $\hat{C} \in \mathbb{R}^{n\times p}$ such that $\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \le \epsilon$, there exists an $P \in \mathbb{R}^{n\times p}$ and orthogonal $U \in \mathbb{R}^{p\times p}$ such that $\hat{C} = CU + P$ and $\|P\|_F \le \epsilon$.*

Proof. Define $U^T = \operatorname{argmin}_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F$. Then by assumption $\left\| C - \hat{C}U^T \right\|_F \le \epsilon$. Further, the set of orthogonal matrices of size 1 or greater is nonempty so such a U must always exist. Now define $\hat{C} = CU + P$. Because we can pick any $P \in \mathbb{R}^{n\times r}$, such a P always exists. Then

$$\begin{aligned}
\|P\|_F &= \|CU - CU - P\|_F \\
&= \left\| CU - \hat{C} \right\|_F \\
&= \left\| C - \hat{C}U^T \right\|_F \\
&\le \epsilon.
\end{aligned}$$

□

Lemma 25. *For C and $\hat{C}$ in $\mathbb{R}^{n\times p}$, let $\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \le \epsilon$ and $\|C\|_F \le \delta$. Then*

$$\left\| CC^T - \hat{C}\hat{C}^T \right\|_F \le 2\delta\epsilon + \epsilon^2.$$

Proof. For any $W : WW^T = I$, we can write

$$\begin{aligned}
\left\| \hat{C} \right\|_F = \left\| \hat{C}W \right\|_F &= \left\| \hat{C}W - C + C \right\|_F \\
&\le \left\| \hat{C}W - C \right\|_F + \|C\|_F \\
&= \left\| C - \hat{C}W \right\|_F + \|C\|_F ,
\end{aligned}$$

where we have made use of triangle inequality for norms and the unitarily invariance of the Frobenius norm. The above holds for any W. Selecting $W^* = \operatorname{argmin}_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F$, we have

$$\begin{aligned}
\left\| \hat{C} \right\|_F &\le \left\| C - \hat{C}W^* \right\|_F + \|C\|_F \\
&\le \epsilon + \delta.
\end{aligned}$$

Again, for any $W : WW^T = I$, we can write

$$\begin{aligned}
\left\| CC^T - \hat{C}\hat{C}^T \right\|_F &= \left\| CC^T - \hat{C}WW^T\hat{C}^T \right\|_F \\
&= \left\| CC^T - \hat{C}WC^T + \hat{C}WC^T - \hat{C}WW^T\hat{C}^T \right\|_F \\
&= \left\| [C - \hat{C}W]C^T + \hat{C}W[C^T - W^T\hat{C}^T] \right\|_F \\
&\le \left\| [C - \hat{C}W]C^T \right\|_F + \left\| \hat{C}W[C^T - W^T\hat{C}^T] \right\|_F \\
&\le \|C\|_F \left\| C - \hat{C}W \right\|_F + \left\| \hat{C}W \right\|_F \left\| C - \hat{C}W \right\|_F \\
&= \|C\|_F \left\| C - \hat{C}W \right\|_F + \left\| \hat{C} \right\|_F \left\| C - \hat{C}W \right\|_F ,
\end{aligned}$$

where in addition to the two properties of the Frobenius norm we used before we have also used the submultiplicative property of this norm. We can again pick W arbitrarily, and we again pick $W^* = \operatorname{argmin}_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F$, so that

$$\begin{aligned}
\left\| CC^T - \hat{C}\hat{C}^T \right\|_F &\leq \|C\|_F \left\| C - \hat{C}W^* \right\|_F + \left\| \hat{C} \right\|_F \left\| C - \hat{C}W^* \right\|_F \\
&\leq \|C\|_F \, \epsilon + \|\hat{C}\|_F \epsilon \\
&\leq \delta\epsilon + (\epsilon + \delta)\epsilon \\
&= 2\delta\epsilon + \epsilon^2,
\end{aligned}$$

where we have used the bound on $\|\hat{C}\|_F$ in moving to the second to last line of the proof. □

Lemma 26. *For $C_1, C_2 \in \mathbb{R}^{n\times p}$, such that $\|C_1\|_F \leq \delta_1$ and $\|C_2\|_F \leq \delta_2$ define*

$$\begin{aligned}
\hat{C}_1 &= C_1 + P_1 \\
\hat{C}_2 &= C_2 + P_2,
\end{aligned}$$

for some error matrices $P_1, P_2 \in \mathbb{R}^{n\times p}$ such that $\|P_1\|_F \leq \epsilon_1$ and $\|P_2\|_F \leq \epsilon_2$. Then

$$\left\| C_1^T C_2 - \hat{C}_1^T \hat{C}_2 \right\|_F \leq \epsilon_1\delta_2 + \epsilon_1\epsilon_2 + \delta_1\epsilon_2.$$

Proof. Throughout this proof we will use the triangle inequality of norms and the submultiplicative property of the Frobenius norm without further comment.

First, note that

$$\begin{aligned}
\left\| \hat{C}_1 \right\|_F &= \left\| \hat{C}_1 - C_1 + C_1 \right\|_F \\
&\leq \left\| \hat{C}_1 - C_1 \right\|_F + \|C_1\|_F \\
&= \|P_1\|_F + \|C_1\|_F \\
&\leq \epsilon_1 + \delta_1.
\end{aligned}$$

Next, we have

$$\begin{aligned}
\left\| C_1^T C_2 - \hat{C}_1^T \hat{C}_2 \right\|_F &= \left\| C_1^T C_2 - \hat{C}_1^T C_2 + \hat{C}_1^T C_2 - \hat{C}_1^T \hat{C}_2 \right\|_F \\
&= \left\| \left[C_1^T - \hat{C}_1^T \right] C_2 + \hat{C}_1^T \left[C_2 - \hat{C}_2 \right] \right\|_F \\
&= \left\| -P_1^T C_2 - \hat{C}_1^T P_2 \right\|_F \\
&\leq \|P_1^T C_2\|_F + \left\| \hat{C}_1^T P_2 \right\|_F \\
&\leq \|P_1\|_F \|C_2\|_F + \|\hat{C}_1\|_F \|P_2\|_F \\
&\leq \epsilon_1\delta_2 + [\epsilon_1 + \delta_1]\epsilon_2 \\
&= \epsilon_1\delta_2 + \epsilon_1\epsilon_2 + \delta_1\epsilon_2,
\end{aligned}$$

where he have used the bound on $\left\| \hat{C}_1 \right\|$ in moving to the second to last line of the proof. □

Lemma 27. *Let U be an $n \times n$ orthogonal matrix. For the unit length vector $v \in \mathbb{R}^n$, define $\theta_i(v) = \min_{k:|k|=1} \cos^{-1}(kU(:,i)^T v)$ where $|\cdot|$ indicates absolute value. Then*

$$\sum_{j \neq i} (U(:,j)^T v)^2 = \sin^2 \theta_i(v).$$

Proof. The matrix U is orthogonal, so it must be that

$$\sum_j (U(:,j)^T v)^2 = v^T U U^T v = v^T v = \|v\|_2^2 = 1,$$

where $\|\cdot\|_2$ indicates the ℓ_2-norm. Therefore, for $k \in \{1, -1\}$, we have

$$\sum_j (U(:,j)^T v)^2 = (kU(:,i)^T v)^2 + \sum_{j \neq i} (U(:,j)^T v)^2 = \cos^2 \theta_i(v) + \sum_{j \neq i} (U(:,j)^T v)^2 = 1,$$

which we can use to find

$$\sum_{j \neq i} (U(:,j)^T v)^2 = 1 - \cos^2 \theta_i(v) = \sin^2 \theta_i(v).$$

□

Lemma 28. *Consider the matrices $A, \tilde{A} \in \mathcal{S}_+^n$ of rank $r > 0$ and $\tilde{r} \geq r$, respectively. Define $\hat{C} \in \mathbb{R}^{n \times r}$ column-wise such that*

$$\hat{C}(:,i) = \tilde{\lambda}_i^{1/2} \tilde{E}(:,i), \qquad \forall i \leq r. \tag{B.6}$$

For $i \leq r$, define $\theta_i = \min_{k_i:|k_i|=1} \cos^{-1}(k_i \tilde{E}_i^T E_i)$, where $|\cdot|$ indicates absolute value. Then for any $C \in \mathbb{R}^{n \times r}$ such that $A = CC^T$ and $\tilde{A}$

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \leq \sqrt{\sum_{i=1}^r |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i \sin^2(\theta_i/2) + 4\lambda_i^{1/2} |\lambda_i - \tilde{\lambda}_i|^{1/2} \sin^2(\theta_i/2)}.$$

Proof. Let $\mathcal{S}$ be the set of $r \times r$ diagonal sign matrices (diagonal matrices with values of 1 or -1 along the diagonal). The Frobenius norm is unitarily invariant, so for any C and $\hat{C}$,

$$\begin{aligned}\min_{W:WW^T=I} \|C - \hat{C}W\|_F &= \min_{W:WW^T=I} \|CW - \hat{C}\|_F \\ &= \min_{W:WW^T=I, S \in \mathcal{S}} \|CWS - \hat{C}S\|_F.\end{aligned}$$

Further, for orthogonal W, the matrix WS^{-1} is also orthogonal, so if W^* minimizes $\min_{W:WW^T=I, S\in\mathcal{S}} \|CW - \hat{C}S\|_F$, then $W^* S^{-1}$ is an orthogonal matrix that minimizes $\min_{W:WW^T=I, S \in \mathcal{S}} \|CWS - \hat{C}S\|_F$, establishing that

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F = \min_{W:WW^T=I, S\in\mathcal{S}} \|CW - \hat{C}S\|_F.$$

Now select W' such that $CW' = E\Lambda^{1/2}$. If $CC^T = A$, such a W' is guaranteed to always exist by lemma 18. Next, note that

$$\begin{aligned}\min_{W:WW^T=I, S\in\mathcal{S}} \|CW - \hat{C}S\|_F &\le \min_{S\in\mathcal{S}} \|CW' - \hat{C}S\|_F \\ &= \min_{S\in\mathcal{S}} \|E\Lambda^{1/2} - \hat{C}S\|_F.\end{aligned}$$

We next can write

$$\begin{aligned}\min_{S\in\mathcal{S}} \|E\Lambda^{1/2} - \hat{C}S\|_F^2 &= \min_{S\in\mathcal{S}} \sum_{i=1}^{r} \|E(:,i)\lambda_i^{1/2} - \hat{C}(:,i)S(i,i)\|_2^2 \\ &= \sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \|E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}\|_2^2, \end{aligned} \tag{B.7}$$

where in moving the last line above we have used (B.6) to substitute for $\hat{C}(:,i)$ and written the minimization over $S \in \mathcal{S}$ equivalently in terms of individual $k_i \in \{-1,1\}$.

Now, let $\tilde{E}_\perp$ be a matrix with orthonormal columns that span the subspace of $\mathbb{R}^n$ orthogonal to the subspace spanned by the first r columns of $\tilde{E}$. The columns of the unitary matrix $B = [\tilde{E}_1, \ldots, \tilde{E}_k, \tilde{E}_\perp]$ then form an orthonormal basis for $\mathbb{R}^n$. For any vector $x \in \mathbb{R}^n$, it must then be that $\|x\|_2 = \|B^T x\|_2$. Therefore,

$$\begin{aligned}\sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \|E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}\|_2^2 &= \sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \|B^T[E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}]\|_2^2 \\ &= \sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \sum_{j=1}^{n} \left(B(:,j)^T[E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}]\right)^2 \\ &= \sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \Bigg(\left(B(:,i)^T[E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}]\right)^2 + \ldots \\ &\qquad \sum_{j\neq i} \left(B(:,j)^T[E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}]\right)^2\Bigg) \\ &= \sum_{i=1}^{r} \min_{k_i:k_i\in\{-1,1\}} \Bigg(\left(\lambda_i^{1/2}\tilde{E}(:,i)^T E(:,i) - k_i\tilde{\lambda}_i^{1/2}\right)^2 + \ldots \\ &\qquad \sum_{j\neq i} \left(\lambda_i^{1/2} B(:,j)^T E(:,i)\right)^2\Bigg),\end{aligned}$$

where in moving to the last equality statement we have recognized that the first r columns of B are by construction the first r columns of $\tilde{E}$.

Using lemma 27, it must be that

$$\begin{aligned}\sum_{j\neq i} \left(\lambda_i^{1/2} B(:,j)^T E(:,i)\right)^2 &= \lambda_i \sum_{j\neq i} \left(B(:,j)^T E(:,i)\right)^2 \\ &= \lambda_i \sin^2\theta_i,\end{aligned}$$

where $\theta_i = \min_{k_i:|k_i|=1} \cos^{-1}(k_i B(:,i)^T E(:,i)) = \min_{k_i:|k_i|=1} \cos^{-1}(k_i\tilde{E}(:,i)^T E(:,i))$.

Further, note that

$$\min_{k_i : k_i \in \{-1,1\}} \left(\lambda_i^{1/2}\tilde{E}(:,i)^T E(:,i) - k_i\tilde{\lambda}_i^{1/2}\right)^2 = \min_{k_i : k_i \in \{-1,1\}} \left(k_i\lambda_i^{1/2}\tilde{E}(:,i)^T E(:,i) - \tilde{\lambda}_i^{1/2}\right)^2.$$

Additionally, $\lambda_i^{1/2} \geq 0$ and $\tilde{\lambda}_i^{1/2} \geq 0$ since both A and $\tilde{A}$ are PSD. Therefore, the k_i that minimizes the above equation must render $k_i\tilde{E}(:,i)^T E(:,i) \geq 0$, from which we can conclude that $k_i\tilde{E}(:,i)^T E(:,i) = \cos(\theta_i)$, when $\theta_i = \min_{k_i:|k_i|=1} \cos^{-1}(k_i\tilde{E}(:,i)^T E(:,i))$, so

$$\min_{k_i : k_i \in \{-1,1\}} \left(k_i\lambda_i^{1/2}\tilde{E}(:,i)^T E(:,i) - \tilde{\lambda}_i^{1/2}\right)^2 = \left(\lambda_i^{1/2}\cos(\theta_i) - \tilde{\lambda}_i^{1/2}\right)^2.$$

We can therefore write (B.7) as

$$\begin{aligned}\sum_{i=1}^{r} \min_{k_i : k_i \in \{-1,1\}} \|E(:,i)\lambda_i^{1/2} - k_i\tilde{E}(:,i)\tilde{\lambda}_i^{1/2}\|_2^2 &= \sum_{i=1}^{r} \left[\left(\lambda_i^{1/2}\cos(\theta_i) - \tilde{\lambda}_i^{1/2}\right)^2 + \lambda_i \sin^2\theta_i\right] \\ &= \sum_{i=1}^{r} \left[\lambda_i + \tilde{\lambda}_i - 2\cos(\theta_i)\lambda_i^{1/2}\tilde{\lambda}_i^{1/2}\right].\end{aligned}$$

We now bound

$$\begin{aligned}\lambda_i + \tilde{\lambda}_i - 2\cos(\theta_i)\lambda_i^{1/2}\tilde{\lambda}_i^{1/2} &= (\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2})^2 + 2\lambda_i^{1/2}\tilde{\lambda}_i^{1/2} - 2\cos(\theta_i)\lambda_i^{1/2}\tilde{\lambda}_i^{1/2} \\ &= (\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2})^2 + 2\lambda_i^{1/2}\tilde{\lambda}_i^{1/2}\left(1 - \cos(\theta_i)\right).\end{aligned}$$

Note that

$$\begin{aligned}(\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2})^2 &= \frac{(\lambda_i - \tilde{\lambda}_i)^2}{(\lambda_i^{1/2} + \tilde{\lambda}_i^{1/2})^2} \\ &= |\lambda_i - \tilde{\lambda}_i| \frac{|\lambda_i - \tilde{\lambda}_i|}{(\lambda_i + \tilde{\lambda}_i + 2\lambda_i^{1/2}\tilde{\lambda}_i^{1/2})} \\ &\leq |\lambda_i - \tilde{\lambda}_i| \frac{|\lambda_i - \tilde{\lambda}_i|}{\lambda_i + \tilde{\lambda}_i} \\ &\leq |\lambda_i - \tilde{\lambda}_i|,\end{aligned}$$

and we can use a half angle formula to write $\cos(\theta_i) = 1 - 2\sin^2(\theta_i/2)$, and we have

$$\lambda_i + \tilde{\lambda}_i - 2\cos(\theta_i)\lambda_i^{1/2}\tilde{\lambda}_i^{1/2} \leq |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i^{1/2}\tilde{\lambda}_i^{1/2}\sin^2(\theta_i/2).$$

Further note that $\tilde{\lambda}_i^{1/2} = \lambda_i^{1/2} + \tilde{\lambda}_i^{1/2} - \lambda_i^{1/2} \leq \lambda_i^{1/2} + |\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2}|$, so we have

$$\begin{aligned}\lambda_i + \tilde{\lambda}_i - 2\cos(\theta_i)\lambda_i^{1/2}\tilde{\lambda}_i^{1/2} &\leq |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i^{1/2}(\lambda_i^{1/2} + |\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2}|)\sin^2(\theta_i/2) \\ &= |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i\sin^2(\theta_i/2) + 4\lambda_i^{1/2}|\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2}|\sin^2(\theta_i/2) \\ &\leq |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i\sin^2(\theta_i/2) + 4\lambda_i^{1/2}|\lambda_i - \tilde{\lambda}_i|^{1/2}\sin^2(\theta_i/2),\end{aligned}$$

where in moving to the last line we have again used the inequality $(\lambda_i^{1/2} - \tilde{\lambda}_i^{1/2})^2 \leq |\lambda_i - \tilde{\lambda}_i|$. We can then complete the proof by writing

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F^2 \leq \sum_{i=1}^{r} |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i \sin^2(\theta_i/2) + 4\lambda_i^{1/2}|\lambda_i - \tilde{\lambda}_i|^{1/2} \sin^2(\theta_i/2),$$

where the final result is obtained by taking the square root of both sides of the above inequality.

□

Lemma 29. *Consider the matrices $A, \tilde{A} \in \mathcal{S}_+^n$ of rank $r > 0$ and $\tilde{r} \geq r$, respectively. Define $S = \tilde{A} - A$. Let $\delta = \min\{\min_{i\in[r-1],} |\lambda_i - \lambda_{i+1}|, \lambda_r\}$. Let $\|S\|_F \leq \epsilon$, for some $\epsilon < \delta/2 \leq 1$. Define $\hat{C} \in \mathbb{R}^{n\times r}$ column-wise as*

$$\hat{C}(:, i) = \tilde{\lambda}_i^{1/2} \tilde{E}(:, i), \qquad \forall i \leq r. \tag{B.8}$$

Then for any C such that $CC^T = A$,

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \leq K\sqrt{r\epsilon},$$

for $K = \sqrt{1 + \frac{16\lambda_1}{\delta^2} + \frac{8\sqrt{2}\lambda_1^{1/2}}{\delta^{3/2}}}$.

Proof. For $i \leq r$, define $\theta_i = \min_{k_i:|k_i|=1} \cos^{-1}(k\tilde{E}_i^T E_i)$. We start by using lemma 28 to write

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \leq \sqrt{\sum_{i=1}^{r} |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i \sin^2(\theta_i/2) + 4\lambda_i^{1/2}|\lambda_i - \tilde{\lambda}_i|^{1/2} \sin^2(\theta_i/2)}. \tag{B.9}$$

We will now proceed to establish our theorem by using results in matrix perturbation theory to bound the different terms in the square root of equation (B.9). Throughout this proof it should be understood that $\lambda_{r+1}, \ldots, \lambda_n = 0$ and $\tilde{\lambda}_{\tilde{r}+1}, \ldots, \tilde{\lambda}_n = 0$).

By construction S is a real, symmetric matrix, and we begin by using Weyl's theorem, which states that $\tilde{\lambda}_i \in [\lambda_i + \lambda_1(S), \lambda_i + \lambda_n(S)]$ [126], where $\lambda_i(S)$ indicates the i^{th} eigenvalue of S, to bound $|\lambda_i - \tilde{\lambda}_i| \leq \max_{j\in\{1,n\}} |\lambda_j(S)| \leq \max_{j\in[$
We then have

$$\sum_{i=1}^{r} |\lambda_i - \tilde{\lambda}_i| \leq \sum_{i=1}^{r} \max_{j\in[n]} |\lambda_j(S)| = r \max_{j\in[n]} |\lambda_j(S)| = r\|S\|_2 \leq r\,\|S\|_F \leq r\epsilon, \tag{B.10}$$

where $\|\cdot\|_2$ indicates the spectral norm.

We next bound $\sin^2(\theta_i/2)$ for $i \leq r$. For $\tilde{E}(:, i)$, we define

$$r_i = A\tilde{E}(:, i) - \tilde{\lambda}_i \tilde{E}(:, i), \tag{B.11}$$

which is a residual vector that will be 0 if and only if $(\tilde{\lambda}_i, \tilde{E}(:, i))$ are an eigenpair of A.

Now define

$$\mathbf{sep}\{A, \tilde{A}\} = \min_{i\in[r]} \min_{j\in[n], j\neq i} |\tilde{\lambda}_i - \lambda_j|. \tag{B.12}$$

By the Sin Theta theorem (c.f., theorem 3.4 in §V of [126]), we know that

$$|\sin(\theta_i)| \leq \frac{\|r_i\|_2}{\min_{j\in[n],j\neq i} |\lambda_j - \tilde{\lambda}_i|}.$$

We also know $\mathbf{sep}\{A, \tilde{A}\} \leq \min_{j\in[n],j\neq i} |\lambda_j - \tilde{\lambda}_i|$, for all $i \in [r]$, so we have

$$\sum_{i=1}^{r} \sin^2(\theta_i) \leq \sum_{i=1}^{r} \left(\frac{\|r_i\|_2}{\mathbf{sep}\{A, \tilde{A}\}}\right)^2 \leq \frac{1}{\mathbf{sep}^2\{A, \tilde{A}\}} \sum_{i=1}^{r} \|r_i\|_2^2.$$

We now bound $\sum_{i=1}^{r} \|r_i\|_2^2$. To simplify things note that

$$r_i = A\tilde{E}(:,i) - \tilde{\lambda}_i\tilde{E}(:,i) = A\tilde{E}(:,i) - \tilde{A}\tilde{E}(:,i) = A\tilde{E}(:,i) - (A+S)\tilde{E}(:,i) = -S\tilde{E}(:,i),$$

so that $\|r_i\|_2 = \left\|S\tilde{E}(:,i)\right\|_2$. We can now write

$$\begin{aligned}\sum_{i=1}^{r} \|r_i\|_2^2 &= \sum_{i=1}^{r} \left\|S\tilde{E}(:,i)\right\|_2^2 \\ &= \left\|S\tilde{E}(:,1:r)\right\|_F^2 \\ &\leq \left\|S[\tilde{E}(:,1:r), \tilde{E}(:,1:r)_\perp]\right\|_F^2 \\ &= \|S\|_F^2 \\ &\leq \epsilon^2,\end{aligned}$$

where $\tilde{E}(:,1:r)_\perp$ is a matrix with orthonormal columns spanning a subspace orthogonal to the subspace spanned by the columns of the matrix $\tilde{E}(:,1:r)$, so $[\tilde{E}(:,1:r), \tilde{E}(:,1:r)_\perp]$ is an orthogonal matrix.

Therefore, we have

$$\sum_{i=1}^{r} \sin^2(\theta_i) \leq \frac{\epsilon^2}{\mathbf{sep}^2\{A, \tilde{A}\}}. \tag{B.13}$$

To complete our bound on $\sum_{i=1}^{r} \sin^2(\theta_i)$, we have only to lower bound $\mathbf{sep}\{A, \tilde{A}\}$. Let $d_{\max} = \max_{i\in[r]} |\lambda_i - \tilde{\lambda}_i|$. We can again use Weyl's theorem to bound

$$d_{\max} = \max_{i\in[r]} |\lambda_i - \tilde{\lambda}_i| \leq \|S\|_2 \leq \|S\|_F \leq \epsilon < \delta/2. \tag{B.14}$$

For $i \in [r]$ we also derive an inequality that will be immediately useful. We start by stating,

$$\min_{j\in[n],j\neq i} |\lambda_i - \lambda_j| = \min\{|\lambda_{i-1} - \lambda_i|, |\lambda_{i+1} - \lambda_i|\},$$

where is understood the appropriate terms disappear if $i = 1$ or $i = n$. We finish the derivation of the inequality by writing for all $i \in [r]$,

$$\min\{|\lambda_{i-1}-\lambda_i|, |\lambda_{i+1}-\lambda_i|\} \le \min\{\min_{i\in[r-1],} |\lambda_i - \lambda_{i+1}|, \lambda_r\} = \delta,$$

so we can conclude for all $i \in [r]$,

$$\min_{j\in[n],j\neq i} |\lambda_i - \lambda_j| \le \delta. \tag{B.15}$$

For all $i \in [r]$, we can then use B.14 and B.15 to bound

$$\begin{aligned}
\min_{j\in[n],j\neq i} |\lambda_j - \tilde{\lambda}_i| &= \min_{j\in[n],j\neq i} |\tilde{\lambda}_i - \lambda_j| \\
&= \min_{j\in[n],j\neq i} |\tilde{\lambda}_i - \lambda_i + \lambda_i - \lambda_j| \\
&\ge \min_{j\in[n],j\neq i} |\lambda_i - \lambda_j| - |\tilde{\lambda}_i - \lambda_i| \\
&\ge \min_{j\in[n],j\neq i} \delta - |\tilde{\lambda}_i - \lambda_i| \\
&\ge \min_{j\in[n],j\neq i} \delta - d_{\text{max}} \\
&> \delta - \delta/2 \\
&= \delta/2.
\end{aligned}$$

Therefore, it must be that $\mathbf{sep}\{A, \tilde{A}\} > \delta/2$. We can then pick up from (B.13) and complete our bound on $\sum_{i=1}^r \sin^2(\theta_i)$ as

$$\sum_{i=1}^r \sin^2(\theta_i) < \frac{4\epsilon^2}{\delta^2}. \tag{B.16}$$

We now pick back up with (B.9) and write

$$\begin{aligned}
\min_{W:WW^T=I} \|C - \hat{C}W\|_F &\le \sqrt{\sum_{i=1}^r |\lambda_i - \tilde{\lambda}_i| + 4\lambda_i \sin^2(\theta_i/2) + 4\lambda_i^{1/2}|\lambda_i - \tilde{\lambda}_i|^{1/2}\sin^2(\theta_i/2)} \\
&= \sqrt{\sum_{i=1}^r |\lambda_i - \tilde{\lambda}_i| + 4\sum_{i=1}^r \lambda_i \sin^2(\theta_i/2) + 4\sum_{i=1}^r \lambda_i^{1/2}|\lambda_i - \tilde{\lambda}_i|^{1/2}\sin^2(\theta_i/2)} \\
&\le \sqrt{\sum_{i=1}^r |\lambda_i - \tilde{\lambda}_i| + 4\lambda_1 \sum_{i=1}^r \sin^2(\theta_i/2) + 4\lambda_1^{1/2} d_{\text{max}}^{1/2} \sum_{i=1}^r \sin^2(\theta_i/2)}.
\end{aligned}$$

Further, note that $\theta_i \in [0, \pi/2]$, and therefore $\sin(\theta_i/2) \le \sin(\theta_i)$, so we have

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \le \sqrt{\sum_{i=1}^r |\lambda_i - \tilde{\lambda}_i| + 4\lambda_1 \sum_{i=1}^r \sin^2(\theta_i) + 4\lambda_1^{1/2} d_{\text{max}}^{1/2} \sum_{i=1}^r \sin^2(\theta_i)}.$$

We now use (B.10), (B.14) and (B.16) to find

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \leq \sqrt{r\epsilon + \frac{16\lambda_1\epsilon^2}{\delta^2} + \frac{8\sqrt{2}\lambda_1^{1/2}\epsilon^2}{\delta^{3/2}}}.$$

For $r \geq 1$ and $\epsilon \leq 1$, we then have

$$\min_{W:WW^T=I} \|C - \hat{C}W\|_F \leq K\sqrt{r\epsilon},$$

for $K = \sqrt{1 + \frac{16\lambda_1}{\delta^2} + \frac{8\sqrt{2}\lambda_1^{1/2}}{\delta^{3/2}}}$.

□

Lemma 30. *Let M be a rank $r > 0$ matrix in $\mathbb{R}^{n\times r}$ such that $\|M\|_F \leq \delta$. Define*

$$M_1 = MW_1 + P_1$$
$$M_2 = MW_2 + P_2,$$

for perturbation matrices $P_1, P_2 \in \mathbb{R}^{n\times r}$ such that $\|P_1\|_F \leq \epsilon_1$ and $\|P_2\|_F \leq \epsilon_2$ and orthonormal matrices $W_1 \in \mathbb{R}^{r\times r}$ and $W_2 \in \mathbb{R}^{r\times r}$.

Find

$$T = \underset{W:WW^T=I}{\operatorname{argmin}} \|MW_1 - MW_2W\|_F^2, \tag{B.17}$$

and

$$\hat{T} = \underset{W:WW^T=I}{\operatorname{argmin}} \|M_1 - M_2W\|_F^2. \tag{B.18}$$

Then $T = W_2^TW_1$ is unique. If $M_2^TM_1$ is also of rank r, $\hat{T}$ is also unique. Finally, when $M_2^TM_1$ is of rank r,

$$\left\|T - \hat{T}\right\|_F \leq \frac{2}{\sigma_r^2(M)}[\delta(\epsilon_1 + \epsilon_2) + \epsilon_1\epsilon_2].$$

Proof. Define $S = [MW_2]^TMW_1$, and denote the singular value decomposition of S as $S = U\Sigma V^T$, for orthogonal matrices U and V and the diagonal matrix Σ containing the singular values of S down its diagonal. Similarly, define $\hat{S} = M_2^TM_1$, and denote its singular value decomposition as $\hat{S} = \hat{U}\hat{\Sigma}\hat{V}^T$.

It has been shown in [95] that the T and $\hat{T}$ which minimize (B.17) and (B.18) can be expressed as

$$T = UV^T \tag{B.19}$$
$$\hat{T} = \hat{U}\hat{V}^T. \tag{B.20}$$

Denote the singular value decomposition of M as $M = L\Psi R^T$, for the matrix with orthornormal columns $L \in \mathbb{R}^{n\times r}$, the orthogonal matrix $R \in \mathbb{R}^{r\times r}$ and the diagonal matrix $\Psi \in \mathbb{R}^{r\times r}$. Writing

$$S = [MW_2]^TMW_1 = W_2^TM^TMW_1 = W_2^TR\Psi L^TL\Psi R^TW_1 = W_2^TR\Psi^2R^TW_1,$$

makes it clear that $U = W_2^T R$ and $V = W_1^T R$. Therefore

$$T = UV^T = W_2^T R R^T W_1 = W_2^T W_1.$$

To establish the uniqueness of T, [95] has shown that it is sufficient that the matrix S have r, non-zero singular values, which must be the case since we assume that the rank of M is r. Similarly, the uniqueness of $\hat{T}$ follows from the assumption that $\hat{S} = M_2^T M_1$ is of rank r.

All that remains is to establish the bound for $\left\| T - \hat{T} \right\|_F$. We first bound $\left\| S - \hat{S} \right\|_F$, and we write

$$\begin{aligned} \left\| S - \hat{S} \right\|_F &= \left\| [MW_2]^T M W_1 - M_2^T M_1 \right\|_F \\ &= \left\| [MW_2]^T M W_1 - [MW_2 + P_2]^T [MW_1 + P_1] \right\|_F . \end{aligned}$$

Note that $\|MW_2\|_F = \|MW_1\|_F = \|M\|_F \leq \delta$. Lemma 26 can now be applied to find

$$\left\| S - \hat{S} \right\|_F \leq \epsilon_1 \delta + \epsilon_2 \epsilon_1 + \epsilon_2 \delta = \delta(\epsilon_1 + \epsilon_2) + \epsilon_1 \epsilon_2. \tag{B.21}$$

Having established a bound on $\|S - \hat{S}\|$, we now bound $\|T - \hat{T}\|$. Finding T and $\hat{T}$ in (B.19) and (B.20) is equivalent to finding the unitary matrix in the polar decomposition of S and $\hat{S}$. Li ([127], theorem 1) has shown if A and $\tilde{A}$ are two non-singular $n \times n$ matrices, then $\left|\left|\left| Q - \tilde{Q} \right|\right|\right| \leq \frac{2}{\sigma_n(A) + \sigma_n(\tilde{A})} \left|\left|\left| A - \tilde{A} \right|\right|\right|$, where Q and $\tilde{Q}$ are the unitary matrices in the polar decomposition of A and $\tilde{A}$, $\sigma_n(\cdot)$ gives the n^{th} singular value of it's argument when singular values are indexed such that $\sigma_1 \geq \ldots \geq \sigma_n$ and $|||\cdot|||$ is any unitarily invariant norm. We can apply this theorem to $S \in \mathbb{R}^{r \times r}$ and $\hat{S} \in \mathbb{R}^{r \times r}$ as we have shown $S = [MW_2]^T M W_1$ is of rank r and by assumption $\hat{S} = M_2^T M_1$ is as well. Applying the theorem for the Frobenius norm, we find

$$\begin{aligned} \left\| T - \hat{T} \right\|_F &\leq \frac{2}{\sigma_r(S) + \sigma_r(\tilde{S})} \left\| S - \tilde{S} \right\|_F & \text{(B.22)} \\ &\leq \frac{2}{\sigma_r(S)} \left\| S - \tilde{S} \right\|_F & \text{(B.23)} \\ &\leq \frac{2}{\sigma_r(S)} [\delta(\epsilon_1 + \epsilon_2) + \epsilon_1 \epsilon_2]. & \text{(B.24)} \end{aligned}$$

Recognizing that $\sigma_r(S) = \sigma_r([MW_2]^T M W_1) = \sigma_r(M^T M) = \sigma_r^2(M)$, we can complete the proof by writing

$$\left\| T - \hat{T} \right\|_F \leq \frac{2}{\sigma_r^2(M)} [\delta(\epsilon_1 + \epsilon_2) + \epsilon_1 \epsilon_2]. \tag{B.25}$$

□

Lemma 31. *For $C \in \mathbb{R}^{n \times r}$ define $\hat{C}_1$ and $\hat{C}_2$ as*

$$\begin{aligned} \hat{C}_1 &= C(\rho_1, :) W_1 + P_1 \\ \hat{C}_2 &= C(\rho_2 :) W_2 + P_2, \end{aligned}$$

for some index sets $\rho_1 \subseteq [n], \rho_2 \subseteq [n]$ with cardinality $|\rho_1| = n_1$ and $|\rho_2| = n_2$, orthogonal matrices $W_1 \in \mathbb{R}^{r \times r}, W_2 \in \mathbb{R}^{r \times r}$ and error matrices $P_1 \in \mathbb{R}^{n_1 \times r}, P_2 \in \mathbb{R}^{n_2 \times r}$.

Denote the intersection of ρ_1 and ρ_2 as $\iota = \rho_1 \cap \rho_2$. Let ϕ_1 and ϕ_2 indicate the rows of $\hat{C}_1$ and $\hat{C}_2$ so that

$$\hat{C}_1(\phi_1,:) = C(\iota,:)W_1 + P_1$$
$$\hat{C}_2(\phi_2,:) = C(\iota,:)W_2 + P_2.$$

Finally, define η_2 such that $\hat{C}_2(\eta_2,:) = C(\rho_2 \setminus \iota,:)W_2 + P_2$.

Algorithm 2 Procrustes Matrix Reconstruction Algorithm $(\hat{C}_1, \hat{C}_2, \rho_1, \rho_2, \phi_1, \phi_2, \iota, \eta_2)$

```
1. $\hat{T} \leftarrow \text{argmin}_{W:WW^T=I} \left\|\hat{C}_1(\phi_1,:) - \hat{C}_2(\phi_2,:)W\right\|_F$
2. $\hat{C}(\rho_1,:) \leftarrow \hat{C}_1$
3. $\hat{C}(\rho_2 \setminus \iota,:) \leftarrow \hat{C}_2(\eta_2,:)\hat{T}$
4. Return $\hat{C}$
```

Assume $\rho_1 \cup \rho_2 = [n]$, $|\iota| \geq r$, $\|C(\rho_2,:)\|_F \leq \delta_2$, $\|P_1\|_F \leq \epsilon_1$ and $\|P_2\|_F \leq \epsilon_2$. Then if $C(\iota,:)$, and $\hat{C}_2(\phi_2,:)^T\hat{C}_1(\phi_1,:)$ are of rank r, algorithm 2 will assign values to all rows of $\hat{C}$ such that

$$\min_{W:WW^T=I} \left\|C - \hat{C}W\right\|_F \leq \epsilon_1 + \epsilon_2 + K[\delta_2^2(\epsilon_1+\epsilon_2) + \delta_2\epsilon_1\epsilon_2],$$

for a constant $K = \frac{2}{\sigma_r^2(C(\iota,:))}$.

Proof. We first show that all rows of $\hat{C}$ will be assigned values. To see this note that in step 1, rows indexed by ρ_1 are assigned values. In step 2, rows indexed by $\rho_2 \setminus \iota$ are assigned values. Finally, $\rho_1 \cup (\rho_2 \setminus \iota) = [n]$, showing that all rows of $\hat{C}$ are assigned values.

To establish the error bound, we begin by using the triangle inequality to write

$$\min_{W:WW^T=I} \left\|C - \hat{C}W\right\|_F \leq \min_{W:WW^T=I} \left\{\left\|C(\rho_1,:) - \hat{C}(\rho_1,:)W\right\|_F + \left\|C(\rho_2 \setminus \iota,:) - \hat{C}(\rho_2 \setminus \iota,:)W\right\|_F\right\}.$$

If we set $W = W_1^T$, we can write this as

$$\min_{W:WW^T=I} \left\|C - \hat{C}W\right\|_F \leq \left\|C(\rho_1,:) - \hat{C}(\rho_1,:)W_1^T\right\|_F + \left\|C(\rho_2 \setminus \iota,:) - \hat{C}(\rho_2 \setminus \iota,:)W_1^T\right\|_F.$$

We can bound the first term as

$$\begin{aligned}
\left\|C(\rho_1,:) - \hat{C}(\rho_1,:)W_1^T\right\|_F &= \left\|C(\rho_1,:) - \hat{C}_1 W_1^T\right\|_F \\
&= \left\|C(\rho_1,:) - [C(\rho_1,:)W_1 + P_1]W_1^T\right\|_F \\
&= \left\|P_1 W_1^T\right\|_F \\
&= \|P_1\|_F \\
&\leq \epsilon_1,
\end{aligned}$$

where we have used that fact that the Frobenius norm is unitarily invariant in moving from the third to fourth lines. In the remainder of this proof we will use this property of the Frobenius norm without further comment.

We again use the triangle inequality to bound the second term as

$$\begin{aligned}\left\|C(\rho_2\setminus\iota,:)-\hat{C}(\rho_2\setminus\iota,:)W_1^T\right\|_F &= \left\|C(\rho_2\setminus\iota,:)-\hat{C}_2(\eta_2,:)\hat{T}W_1^T\right\|_F\\ &= \left\|C(\rho_2\setminus\iota,:)-[C(\rho_2\setminus\iota,:)W_2+P_2(\eta_2,:)]\hat{T}W_1^T\right\|_F\\ &\le \left\|P_2(\eta_2,:)\hat{T}W_1^T\right\|_F+\left\|C(\rho_2\setminus\iota,:)\left[I-W_2\hat{T}W_1^T\right]\right\|_F\end{aligned}$$

We can bound $\left\|P_2(\eta_2,:)\hat{T}W_1^T\right\|_F$ as

$$\left\|P_2(\eta_2,:)\hat{T}W_1^T\right\|_F=\|P_2(\eta_2,:)\|_F\le\|P_2\|_F\le\epsilon_2,$$

where we have taken advantage of the fact that $\hat{T}W_1^T$ must be an orthogonal matrix. We now complete the proof by bounding $\left\|C(\rho_2\setminus\iota,:)\left[I-W_2\hat{T}W_1^T\right]\right\|_F$. We can use the submultiplicative property of Frobenius norm to write

$$\begin{aligned}\left\|C(\rho_2\setminus\iota,:)\left[I-W_2\hat{T}W_1^T\right]\right\|_F &\le \|C(\rho_2\setminus\iota,:)\|_F\left\|I-W_2\hat{T}W_1^T\right\|_F\\ &= \|C(\rho_2\setminus\iota,:)\|_F\left\|W_2^TW_1-\hat{T}\right\|_F\\ &= \|C(\rho_2\setminus\iota,:)\|_F\left\|T-\hat{T}\right\|_F,\end{aligned}$$

where we have defined $T=W_2^TW_1$. We now apply lemma 30 to bound $\left\|T-\hat{T}\right\|_F$. We set M in the lemma to $C(\iota,:)$, M_1 to $\hat{C}_1(\phi_1,:)$, M_2 to $\hat{C}_2(\phi_2,:)$, P_1 to $P_1(\phi_1,:)$ and P_2 to $P_2(\phi_2,:)$.

Note that it must be that $\|C(\iota,:)\|_F\le\|C(\rho_2,:)\|_F$, any by assumption $\|C(\rho_2,:)\|_F\le\delta_2$ so $\|C(\iota,:)\|_F\le\delta_2$. Additionally, $\|P_1(\phi_1,:)\|_F\le\|P_1\|_F\le\epsilon_1$ and $\|P_2(\phi_2,:)\|_F\le\|P_2\|_F\le\epsilon_2$. By assumption $C(\iota,:)$ is of rank r, as is $\hat{C}_2(\phi_2,:)^T\hat{C}_1(\phi_1,:)$. Under these conditions T is the solution to equation (B.17) in the lemma and $\hat{T}$ to equation (B.18). Applying the lemma, we find

$$\left\|T-\hat{T}\right\|_F\le\frac{2}{\sigma_r^2(C(\iota,:))}[\delta_2(\epsilon_1+\epsilon_2)+\epsilon_1\epsilon_2].$$

We can also bound $\|C(\rho_2\setminus\iota,:)\|_F\le\delta_2$, and we have

$$\begin{aligned}\|C(\rho_2\setminus\iota,:)\|_F\left\|T-\hat{T}\right\|_F &\le \frac{2}{\sigma_r^2(C(\iota,:))}[\delta_2(\epsilon_1+\epsilon_2)+\epsilon_1\epsilon_2]\,\|C(\rho_2\setminus\iota,:)\|_F\\ &\le \frac{2}{\sigma_r^2(C(\iota,:))}[\delta_2(\epsilon_1+\epsilon_2)+\epsilon_1\epsilon_2]\delta_2\\ &= \frac{2}{\sigma_r^2(C(\iota,:))}[\delta_2^2(\epsilon_1+\epsilon_2)+\delta_2\epsilon_1\epsilon_2].\end{aligned}$$

Putting all of the pieces of the proof together we have

$$\min_{W:WW^T=I}\left\|C-\hat{C}W\right\|_F\le\epsilon_1+\epsilon_2+K[\delta_2^2(\epsilon_1+\epsilon_2)+\delta_2\epsilon_1\epsilon_2],$$

for $K=\frac{2}{\sigma_r^2(C(\iota,:))}$.

□

Corollary 32. *For $C \in \mathbb{R}^{n\times r}$, assume there exists a $\hat{C}_1 \in \mathbb{R}^{n_1\times r}$ and $\hat{C}_2 \in \mathbb{R}^{n_2\times r}$ and index sets ρ_1 and ρ_2 such that*

$$\min_{W:WW^T=I} \left\| C(\rho_1,:) - \hat{C}_1 W \right\|_F \leq \epsilon_1$$
$$\min_{W:WW^T=I} \left\| C(\rho_2,:) - \hat{C}_2 W \right\|_F \leq \epsilon_2.$$

Assume $\rho_1 \cup \rho_2 = [n]$. Let $\iota = \rho_1 \cap \rho_2$ and assume $|\iota| \geq r$.

Define $C_l = C(\rho_l,:)$ and let ϕ_l be an index set that assigns the rows of the matrix $C(\iota,:)$ to their location in C_l, so that $C(\iota,:) = C_l(\phi_l,:)$ and let η_l assign the rows of $C(\rho_l \setminus \iota_l,:)$ to their location in C_l, so that $C(\rho_l \setminus \iota_l,:) = C_l(\eta_l,:)$.

Assume $C(\iota,:)$ and $\hat{C}_1(\phi_1,:)^T \hat{C}_2(\phi_2,:)$ each are of rank r. Finally, assume $\|C(\rho_2,:)\|_F \leq \delta_2$.

Learn $\hat{C} \in \mathbb{R}^{n\times r}$ with algorithm 2 in lemma 31. Then algorithm 2 will assign values to all rows of $\hat{C}$ such that

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \leq \epsilon_1 + \epsilon_2 + K[\delta_2^2(\epsilon_1+\epsilon_2) + \delta_2\epsilon_1\epsilon_2],$$

for $K = \frac{2}{\sigma_r^2(C(\iota,:))}$.

Proof. For the $r \times r$ orthogonal matrices T_1 and T_2, define the matrices $E_1, E_2 \in \mathbb{R}^{n\times r}$ such that

$$\hat{C}_1 = C(\rho_1,:)T_1 + E_1 \tag{B.26}$$
$$\hat{C}_2 = C(\rho_2,:)T_2 + E_2. \tag{B.27}$$

Then for any $\hat{C}_1$, $\hat{C}_2$ and C, there exists orthogonal T_1 and T_2 such that $\|E_1\|_F \leq \epsilon_1$ and $\|E_2\|_F \leq \epsilon_2$. We prove this first for $\hat{C}_1$ and $C(\rho_1,:)$. Let $W' = \operatorname{argmin}_{W:WW^T=I} \left\| C(\rho_1,:) - \hat{C}_1 W \right\|_F$, and let $T_1 = W'^T$. Then

$$\begin{aligned}\|E_1\|_F &= \left\| \hat{C}_1 - C(\rho_1,:)W'^T \right\|_F \\ &= \left\| C(\rho_1,:) - \hat{C}_1 W' \right\|_F \\ &= \min_{W:WW^T=I} \left\| C(\rho_1,:) - \hat{C}_1 W \right\|_F \\ &\leq \epsilon_1.\end{aligned}$$

The same argument can be carried out for $\hat{C}_2$, $C(\rho_2,:)$ and T_2. All of the assumptions are then met for lemma 31 to guarantee that algorithm 2 will learn a $\hat{C}$ from $\hat{C}_1$ and $\hat{C}_2$ such that all rows of $\hat{C}$ are assigned and

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \leq \epsilon_1 + \epsilon_2 + K[\delta_2^2(\epsilon_1+\epsilon_2) + \delta_2\epsilon_1\epsilon_2],$$

for $K = \frac{2}{\sigma_r^2(C(\iota,:))}$.

□

Corollary 33. *For $C \in \mathbb{R}^{n\times r}$, assume there exists a set of two or more matrices $\hat{C}_1 \in \mathbb{R}^{n_1\times r}, \ldots, \hat{C}_k \in \mathbb{R}^{n_k\times r}$, and collection of index sets $\rho_1, \ldots, \rho_k$ such that $\|C(\rho_l,:)\|_F \leq b$ for some $b > 0$ and $\min_{W:WW^T=I} \left\| C(\rho_l,:) - \hat{C}_l W \right\|_F \leq \epsilon/b$ some $0 \leq \epsilon \leq b$ for all l.*

Assume $\cup_{l=1}^k \rho_l = [n]$. For all $l \geq 2$, define $\iota_l = \rho_l \cap \left(\cup_{j=1}^{l-1} \rho_j\right)$ and assume $|\iota_l| \geq r$.

Define $C_l = C(\rho_l,:)$ for all l, and let ϕ_l be an index set that assigns the rows of the matrix $C(\iota,:)$ to their location in C_l, so that $C(\iota,:) = C_l(\phi_l,:)$ and let η_l assign the rows of $C(\rho_l \setminus \iota_l,:)$ to their location in C_l, so that $C(\rho_l \setminus \iota_l,:) = C_l(\eta_l,:)$.

Algorithm 3 Sequential Procrustes Matrix Recovery $(\hat{C}_1, \ldots, \hat{C}_k, \rho_1, \ldots, \rho_k, \{\iota_l, \phi_l, \eta_l\}_{l=2}^k)$

```
Initialize $\hat{C}$ as a $n \times r$ matrix.
  1. $\hat{C}(\rho_1, :) \leftarrow \hat{C}_1$
  2. For $l \in \{2, \ldots, k\}$
      (a) $\hat{W}_l \leftarrow \text{argmin}_{WW^T=I} \left\| \hat{C}(\iota_l, :) - \hat{C}_l(\phi_l, :)W \right\|_F^2$
      (b) $\hat{C}(\rho_l \setminus \iota_l, :) \leftarrow \hat{C}_l(\eta_l, :)\hat{W}_l$
  3. Return $\hat{C}$
```

Use algorithm 3 to learn a $\hat{C} \in \mathbb{R}^{n \times r}$. Then if $C(\iota_l, :)$ and $\hat{C}_l(\phi_l, :)^T\hat{C}(\iota_l, :)$ each are of rank r for $l \in 2, \ldots, k$, algorithm 3 will assign values to all the rows of $\hat{C}$ such that

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\| \leq [4 + 12/v]^{k-1}\epsilon,$$

for a constant where $v \leq \sigma_r^2(C(\iota_l, :))/b^2$ for all $l \geq 2$.

Proof. We will first establish a bound assuming $\|C(\rho_l, :)\|_F \leq 1$ for all l and $\epsilon \leq 1$, and will then generalize this result when $\|C(\rho_l, :)\|_F \leq b$ for some $b > 0$ for all l and $\epsilon \leq b$.

We establish the result by induction. We prove the base case for $k = 2$. Note for the case $k = 2$, algorithm 3 will consist of step 1, followed by one iteration of step 2, followed by step 3, which is equivalent to running algorithm 2 of lemma 31 with inputs $\hat{C}_1, \hat{C}_2, \rho_1, \rho_2, \phi_1, \phi_2, \iota_2$ and η_2, where we have listed inputs in the same order as in the heading for algorithm 2. In this case, $\hat{C}_1$ and $\hat{C}_2$ are such that $\min_{W:WW^T=I} \left\| C(\rho_1, :) - \hat{C}_1 W \right\| \leq \epsilon$ and $\min_{W:WW^T=I} \left\| C(\rho_2, :) - \hat{C}_2 W \right\| \leq \epsilon$. Further, by assumption $\rho_1 \cup \rho_2 = [n]$, $|\iota_2| \geq r$ and $C(\iota_2, :)$ is of rank r. If $\hat{C}_2(\phi_2, :)^T\hat{C}_1(\phi_1, :)$ is of rank r, corollary 32 then guarantees that algorithm 2 will learn a $\hat{C}$ such that

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \leq 2\epsilon + K[2\delta_2^2\epsilon + \delta_2\epsilon^2], \tag{B.28}$$

for $K = \frac{2}{\sigma_r^2(C(\iota_2, :))}$ where $\delta_2 = \|C(\rho_2, :)\|_F \leq 1$. Noting that by assumption $\epsilon \leq 1$ and $\sigma_r^2(C(\iota_2, :)) \leq v$, we have

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \leq [2 + 3G]\epsilon \leq [4 + 6G]\epsilon, \tag{B.29}$$

where $G = 2/v$.

We now prove the induction step. When $k \geq 3$, algorithm 3 will consist of running step 1, followed by $k - 1$ iterations of step 2, followed by running step 3. Let $\mathcal{I}_{j-1} = \cup_{l=1}^{j-1}\rho_l$. On the $(j-1)^{th}$ iteration of step 2, assume that the previous steps of the algorithm have learned the rows of $\hat{C}$ indexed by $\mathcal{I}_{j-1}$ (that is $C(\mathcal{I}_{j-1}, :)$) such that $\min_{W:WW^T=I} \left\| C(\mathcal{I}_{j-1}, :) - \hat{C}(\mathcal{I}_{j-1}, :)W \right\| \leq [4 + 6G]^{j-2}\epsilon$. For clarity of what is to follow, define $\epsilon'_{j-1} = [4 + 6G]^{j-2}\epsilon$. Performing the $(j-1)^{th}$ iteration of step 2 of algorithm 3 with $\hat{C}_j$ and the $\hat{C}(\mathcal{I}_{j-1}, :)$ from the previous steps of the algorithm and then performing step 3 is equivalent to running algorithm 2 with the inputs $\hat{C}(\mathcal{I}_{j-1}, :), \hat{C}_j, \mathcal{I}_{j-1}, \rho_j, \phi'_{j-1}, \phi_j, \eta_j, \iota_j$, where we have again listed inputs in the same order as in the heading for algorithm 3, and $\phi'_{j-1} = \iota_j$.

By assumption $\min_{W:WW^T=I} \left\| C(\mathcal{I}_{j-1}, :) - \hat{C}'_{j-1}W \right\|_F \leq \epsilon'_{j-1}$ and $\min_{W:WW^T=I} \left\| C(\rho_j, :) - \hat{C}_j W \right\|_F \leq \epsilon$. Further, $\rho_j \cup \mathcal{I}_{j-1} = \rho_j \cup \left(\cup_{l=1}^{j-1} \rho_l \right) = [n_j]$, where n_j is the number of rows in $\hat{C}(\mathcal{I}_j, :)$. Further, $|\iota_j| \geq r$ and $C(\iota_j, :)$ is of rank r. Then if $\hat{C}_j(\phi_j, :)^T\hat{C}(\phi'_{j-1}, :) = \hat{C}_j(\phi_j, :)^T\hat{C}(\iota_j, :)$ also is of rank r, corollary 32 guarantees that algorithm 2 will learn a full $\hat{C}(\mathcal{I}_j, :)$ from $\hat{C}_j$ and $\hat{C}'_{j-1}$ such that

$$\min_{W:WW^T=I} \left\| C(\mathcal{I}_j,:) - \hat{C}(\mathcal{I}_j,:)W \right\|_F \leq \epsilon + \epsilon'_{j-1} + K[\delta_j^2(\epsilon + \epsilon'_{j-1}) + \delta_j \epsilon'_{j-1}\epsilon],$$

where $\delta_j = \|C(\rho_j,:)\|_F \leq 1$. Again, $\epsilon \leq 1$, so we have

$$\min_{W:WW^T=I} \left\| C(\mathcal{I}_j,:) - \hat{C}(\mathcal{I}_j,:)W \right\|_F \leq \epsilon + \epsilon'_{j-1} + K[\epsilon + 2\epsilon'_{j-1}].$$

Recognizing that again $K = \frac{2}{\sigma_r^2(C(\iota_j,:))} \leq \frac{2}{v} = G$, we further have

$$\min_{W:WW^T=I} \left\| C(\mathcal{I}_j,:) - \hat{C}(\mathcal{I}_j,:)W \right\|_F \leq \epsilon + \epsilon'_{j-1} + G[\epsilon + 2\epsilon'_{j-1}]. \tag{B.30}$$

We can bound the right hand side of (B.30) as

$$\begin{aligned}
\min_{W:WW^T=I} \left\| C(\mathcal{I}_j,:) - \hat{C}(\mathcal{I}_j,:)W \right\|_F &\leq \epsilon + [4+6G]^{j-2}\epsilon + G[\epsilon + 2[4+6G]^{j-2}\epsilon] \\
&= [1+G]\epsilon + [1+2G][4+6G]^{j-2}\epsilon \\
&= [1+G]\epsilon + 2^{j-2}[1+2G][2+3G]^{j-2}\epsilon \\
&\leq [2+3G]\epsilon + 2^{j-2}[2+3G]^{j-1}\epsilon \\
&= \left([2+3G] + 2^{j-2}[2+3G]^{j-1}\right)\epsilon \\
&\leq \left([2+3G]^{j-1} + 2^{j-2}[2+3G]^{j-1}\right)\epsilon \\
&= [1+2^{j-2}][2+3G]^{j-1}\epsilon \\
&\leq [2^{j-2}+2^{j-2}][2+3G]^{j-1}\epsilon \\
&= 2^{j-1}[2+3G]^{j-1}\epsilon \\
&= [4+6G]^{j-1}\epsilon.
\end{aligned} \tag{B.31}$$

Substituting $G = 2/v$, we can conclude

$$\min_{W:WW^T=I} \left\| C(\mathcal{I}_j,:) - \hat{C}(\mathcal{I}_j,:)W \right\|_F \leq [4 + 12/v]^{j-1}\epsilon$$

when $\|C(\rho_j,:)\|_F \leq 1$ for all l and $\epsilon \leq 1$. Finally, since $\mathcal{I}_k = \cup_{l=1}^k \rho_k = [n]$, after processing all k submatrices, algorithm 3 will learn a full $\hat{C}$ such that

$$\min_{W:WW^T=I} \left\| C - \hat{C}W \right\|_F \leq [4 + 12/v]^{k-1}\epsilon.$$

We now generalize this result when $\|C(\rho_l,:)\|_F \leq b$ for some $b > 0$ for all l and $0 \leq \epsilon \leq b$. Define $C' = C/b$ and $\hat{C'}_l = \hat{C}_l/b$ for all l. Then

$$\|C'(\rho_l,:)\|_F = \|C(\rho_l,:)/b\|_F = \frac{1}{b}\|C(\rho_l,:)\|_F \leq 1.$$

Further, since $\min_{W:WW^T=I} \left\| C(\rho_l,:) - \hat{C}_l W \right\|_F \leq \epsilon$ and $\epsilon \leq b$, then

$$\begin{aligned}\min_{W:WW^T=I}\left\|C'(\rho_l,:)-\hat{C'}_l W\right\|_F &= \min_{W:WW^T=I}\left\|\frac{1}{b}C(\rho_l,:)-\frac{1}{b}\hat{C}_l W\right\|_F \\ &= \frac{1}{b}\left[\min_{W:WW^T=I}\left\|C(\rho_l,:)-\hat{C}_l W\right\|_F\right] \\ &\leq \frac{\epsilon}{b} \\ &\leq 1.\end{aligned}$$

for all l. For clarity of what is to follow define $\epsilon' = \epsilon/b$.

Finally, if $v \leq \sigma_r^2(C(\iota_l,:))$ for all $l \geq 2$, then $v' = v/b^2$ will satisfy $v' \leq \sigma_r^2(C'(\iota_l,:))$ for all $l \geq 2$, as can be seen by writing

$$\frac{v}{b^2} \leq \frac{1}{b^2}\sigma_r^2(C(\iota_l,:)) = \sigma_r^2(C(\iota_l,:)/b) = \sigma_r^2(C'(\iota_l,:)).$$

Having established this basic inequalities, it can be verified that using algorithm 3 to reconstruct C from $\hat{C}_1,\ldots,\hat{C}_k$ is equivalent to first using algorithm 3 to reconstruct the rescaled C' from $\hat{C'}_1,\ldots,\hat{C'}_k$ and then multiplying the final $\hat{C'}$ by b to estimate $\hat{C} = b\hat{C'}$. However, we have just shown that $\|C'(\rho_l,:)\|_F \leq 1$ for all l and $\epsilon' \leq 1$, so we can use the result we just established to conclude that

$$\min_{W:WW^T=I}\left\|C'-\hat{C'}W\right\|_F \leq [4+12/v']^{k-1}\epsilon'$$

for $v' \leq \sigma_r^2(C'(\iota_l,:))$ for all l. Using the relations above, we can conclude

$$\min_{W:WW^T=I}\left\|C'-\hat{C'}W\right\|_F \leq [4+12/v']^{k-1}\frac{\epsilon}{b}$$

when for $v' \leq \frac{1}{b^2}\sigma_r^2(C(\iota_l,:))$ for all l.

However, if $\min_{W:WW^T=I}\left\|C'-\hat{C'}W\right\|_F \leq [4+12/v']^{k-1}\frac{\epsilon}{b}$, then

$$\begin{aligned}b\left[\min_{W:WW^T=I}\left\|C'-\hat{C'}W\right\|_F\right] &= \min_{W:WW^T=I}\left\|bC'-b\hat{C'}W\right\|_F \\ &= \min_{W:WW^T=I}\left\|C-\hat{C}W\right\|_F \\ &\leq b[4+12/v']^{k-1}\frac{\epsilon}{b} \\ &= [4+12/v']^{k-1}\epsilon,\end{aligned}$$

so $\min_{W:WW^T=I}\left\|C-\hat{C}W\right\|_F \leq [4+12/v']^{k-1}\epsilon$ which completes the proof.

□

Appendix C

Appendix for Factor Analysis Models and Stitching

C.1 Derivation of the EM Algorithm for the Stitching Scenario

C.1.1 The Model

Under the standard factor analysis (FA) model [98] a high-dimensional vector $y_t \in \mathbb{R}^k$ for trial t is generated according to

$$y_t = Cl_t + \mu + \epsilon_t \tag{C.1}$$

$$l_t \sim \mathcal{N}(0, I) \tag{C.2}$$

$$\epsilon_t \sim \mathcal{N}(0, \Psi), \tag{C.3}$$

where $l_t \in \mathbb{R}^m$ is a vector of latent variables distributed according to a standard multivariate Gaussian distribution, $\mu \in \mathbb{R}^k$ is a constant vector and $\epsilon_t \in \mathbb{R}^k$ models uncorrelated Gaussian noise in the observed variables so that Ψ is a diagonal matrix.

In this work, we augment this model to allow for a subset of the individual entries of y_t to be observed each trial. Specifically, we model

$$z_t = S_t y_t, \tag{C.4}$$

where $z_t \in \mathbb{R}^{n_t}$ contains the subset of n_t observed variables for trial t and $S_t \in \mathbb{R}^{n_t \times k}$ is a selection matrix with zeros and ones along its main diagonal.

C.1.2 A Note on Indexing Notation

At times it will be necessary to index into y_t and z_t. In the rest of this work, we will introduce the notation that $y_t(i)$ indexes the variable in position i of y_t and $z_t(i)$ indexes the corresponding variable in z_t so that $z_t(i) = y_t(i)$. When indexing all other variables, indexes should be understood to reference absolute position within the vector or matrix of interest. We note that when necessary, we will also use MATLAB notation when indexing rows or columns of a matrix so $A(i,:)$ indexes the i^{th} row of matrix A and $A(:,i)$ indexes the i^{th} column.

C.1.3 Learning the Model Parameters

We seek to learn the model parameters, that is μ, C and Ψ, from a set of observed data, $z = \{z_1, \ldots, z_T\}$, of observations. We first show that μ can be learned by a direct maximization of the log-likelihood of z. Then we derive an EM algorithm for learning C and Ψ.

Learning μ

We begin by showing that μ can be learned by direct maximization of the log-likelihood of $\{z\}$. It can be shown that $z_j \perp z_k$ for all $j \neq k$, which allows us to write

$$\log P(\{z\}) = \log \prod_{t=1}^{T} P(z_t) = \sum_{t=1}^{T} \log P(z_t) \tag{C.5}$$

From eqs. C.1-C.4, we derive

$$P(z_t) \sim \mathcal{N}\left(S_t\mu, S_t(CC^T + \Psi)S_t^T\right). \tag{C.6}$$

We can form an objective from eqs. C.5 and C.6. Dropping terms that do not depend on μ we have

$$g(\mu) = \sum_{t=1}^{T} -\frac{1}{2}(z_t - S_t\mu)^T (S_t(CC^T + \Psi)S_t^T)^{-1}(z_t - S_t\mu). \tag{C.7}$$

For ease we will work with the differential (see [128] for a good reference) of eq. C.7. Let $a(\mu, \Delta\mu)$ be the differential of eq. C.7. We have

$$a(\mu, \Delta\mu) = \sum_{t=1}^{T} \Delta\mu^T S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1} z_t - \Delta\mu^T S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1} S_t\mu,$$

from which we can identify

$$\nabla\mu = \sum_{t=1}^{T} S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1}(z_t - S_t\mu). \tag{C.8}$$

We now pause to establish an identity that we will need in the rest of the derivation. Assume that $(S_t(CC^T + \Psi)S_t^T)$ is invertible. Then we have

$$I = S_t(CC^T + \Psi)S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1}.$$

Now, let S_t^* be a diagonal matrix such that $S_t^*(i,i) = 1/S_t(i,i)$. Then

$$S_t^* = S_t^* S_t(CC^T + \Psi)S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1} = (CC^T + \Psi)S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1}.$$

We now proceed to find the MLE estimate for μ by setting the left hand side of eq. C.8 to zero.

$$\begin{aligned} 0 &= \sum_{t=1}^{T} S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1}(z_t - S_t\mu) \\ &= \sum_{t=1}^{T} (CC^T + \Psi)S_t^T (S_t(CC^T + \Psi)S_t^T)^{-1}(z_t - S_t\mu) \\ &= \sum_{t=1}^{T} S_t^*(z_t - S_t\mu) \end{aligned}$$

From here it can be verified that $\nabla \mu = 0$ when

$$\hat{\mu}(i) = \frac{\sum_{t:I_t(i)=1}^{T} z_t(i)}{\sum_{t=1}^{T} I_t(i)}, \tag{C.9}$$

where $I_t(i) = 1$ if and only if variable i is observed on trial t. Further, note that the Hessian of g with respect to μ is $\sum_{t=1}^{T} -S_t^T(S_t(CC^T + \Psi)S_t^T)^{-1}S_t$, which by construction is a negative semidefinite matrix. This certifies that $g(\mu)$ is concave and therefore eq. C.9 globally maximizes eq. C.7.

An EM Algorithm for Learning C and Ψ

Equation C.9 can be used to learn an estimate $\hat{\mu}$ for μ. When y_t is fully observed expectation-maximization (EM) [75] is a standard technique for learning C and Ψ. Here we derive an EM algorithm for the augmented model.

The E-Step

We seek the distribution for $l_t|z_t$. Consider the vector $\begin{bmatrix} l_t \\ z_t \end{bmatrix}$, which will be normally distributed. We calculate the mean of the joint distribution as

$$\begin{aligned}
&\mathbb{E}(l_t) = 0 \\
&\mathbb{E}(z_t) = \mathbb{E}(S_t y_t) = S_t \mathbb{E}(y_t) = S_t \mu.
\end{aligned}$$

The submatrices of the covariance matrix can be calculated as

$$\mathbb{E}\left(l_t l_t^T\right) = I$$

$$\begin{aligned}
\mathbb{E}\left((z_t - S_t\mu)(z_t - S_t\mu)^T\right) &= \mathbb{E}\left((S_t(Cl_t + \epsilon_t))(S_t(Cl_t + \epsilon_t))^T\right) \\
&= \mathbb{E}\left(S_t C l_t l_t^T C^T S_t^T\right) + \mathbb{E}\left(S_t C l_t \epsilon_t^T S_t^T\right) + \mathbb{E}\left(S_t \epsilon_t l_t^T C^T S_t^T\right) + \mathbb{E}\left(S_t \epsilon_t \epsilon_t^T S_t^T\right) \\
&= \mathbb{E}\left(S_t C l_t l_t^T C^T S_t^T\right) + \mathbb{E}\left(S_t \epsilon_t \epsilon_t^T S_t^T\right) \\
&= S_t C \mathbb{E}\left(l_t l_t^T\right) C^T S_t^T + S_t \mathbb{E}\left(\epsilon_t \epsilon_t^T\right) S_t^T \\
&= S_t C C^T S_t^T + S_t \Psi S_t^T \\
&= S_t[CC^T + \Psi]S_t^T
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}\left(l_t(z_t - S_t\mu)^T\right) &= \mathbb{E}\left(l_t(S_t(Cl_t + \epsilon_t))^T\right) \\
&= \mathbb{E}\left(l_t l_t^T C^T S_t^T + l_t \epsilon_t^T S_t^T\right) \\
&= \mathbb{E}\left(l_t l_t^T\right) C^T S_t^T \\
&= C^T S_t^T.
\end{aligned}$$

In the above derivations, we have used the relations $\mathbb{E}(l_t\epsilon_t^T) = 0$, $\mathbb{E}(l_t l_t^T) = I$ and $\mathbb{E}(\epsilon_t\epsilon_t^T) = \Psi$.

We can now use the above calculations to write

$$\begin{bmatrix} l_t \\ z_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ S_t\mu \end{bmatrix}, \begin{bmatrix} I & C^T S_t^T \\ S_t C & S_t(CC^T + \Psi)S_t^T \end{bmatrix}\right) \tag{C.10}$$

In (C.10) notice that $S_t\mu$ is simply a vector consisting of the entries of μ corresponding to the entries of y_t that were observed on trial t. Similarly, S_tC produces a matrix consisting only of the rows of C corresponding to observed variables for trial t and $S_t\Psi S_t^T$ is a diagonal matrix with the variances of the observed variables for trial t down it's diagonal.

Using standard properties of the multivariate Gaussian (e.g., see the appendix of [81]), we can write

$$l_t|z_t \sim \mathcal{N}\left(C^T S_t^T B_t^{-1}(z_t - S_t\mu), I - C^T S_t^T B_t^{-1} S_t C\right), \tag{C.11}$$

where $B_t = S_t(CC^T + \Psi)S_t^T$.

The M-Step

In the M-step, we seek to maximize

$$\mathbb{E}\left(\log P(z, \ell|\Theta)\right), \tag{C.12}$$

with respect to $\Theta = \{C, \mu, \Psi\}$. Above, we have shown that eq. C.9 can be used to learn a $\hat{\mu}$ that globally maximizes the log-likelihood of the data. Further, eq. C.9 is invariant with respect to C and Ψ, so in the work that follows we will maximize eq. C.12 for a fixed $\hat{\mu}$ that has been learned from eq. C.9.

In (C.12) we have introduced the notation $\ell = \{l_1, \ldots, l_T\}$ to indicate the set of latents produced for all observations. For each M-step, the expectation is taken with respect to the posterior distribution of latents calculated in the previous E-step. For ease of notation, we introduce the notation $< \cdot >$ to indicate the expectation with respect to this posterior distribution, and will only explicitly write Θ where clarity requires.

We decompose (C.12) as

$$\langle \log P(z, \ell) \rangle = \langle \log P(z|\ell) \rangle + \langle \log P(\ell) \rangle .$$

The second term is constant with respect to Θ, and for the first we derive

$$
\begin{aligned}
f(\Theta) &= \langle \log P(z|\ell;\Theta)\rangle \\
&= \left\langle \log \prod_{t=1}^{T} P(z_t|l_t) \right\rangle \\
&= \sum_{t=1}^{T} \langle \log P(z_t|l_t)\rangle \\
&= \sum_{t=1}^{T} \left\langle -\frac{1}{2}\left[z_t - S_t(Cl_t+\hat{\mu})\right]^T \left[S_t\Psi S_t^T\right]^{-1}\left[z_t - S_t(Cl_t+\hat{\mu})\right] - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\mathbf{det}\left\{S_t\Psi_t S_t^T\right\}\right)\right\rangle \\
&= \sum_{t=1}^{T} \left\langle -\frac{1}{2} z_t^T [S_t\Psi S_t^T]^{-1} z_t + z_t^T [S_t\Psi S_t^T]^{-1} S_t(Cl_t+\hat{\mu}) - \frac{1}{2}(Cl_t+\hat{\mu})^T S_t^T [S_t\Psi S_t^T]^{-1} S_t(Cl_t+\hat{\mu}) \dots \right. \\
&\qquad\qquad \left. -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\mathbf{det}\left\{S_t\Psi_t S_t^T\right\}\right)\right\rangle \\
&= \sum_{t=1}^{T} \left\langle -\frac{1}{2} z_t^T [S_t\Psi S_t^T]^{-1} z_t + z_t^T [S_t\Psi S_t^T]^{-1} S_t(Cl_t+\hat{\mu}) - \frac{1}{2} l_t^T C^T S_t^T [S_t\Psi S_t^T]^{-1} S_t C l_t \dots \right. \\
&\qquad\qquad \left. -\hat{\mu}^T S_t^T [S_t\Psi S_t^T]^{-1} S_t C l_t - \frac{1}{2}\hat{\mu}^T S_t^T [S_t\Psi S_t^T]^{-1} S_t \hat{\mu} - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\mathbf{det}\left\{S_t\Psi_t S_t^T\right\}\right)\right\rangle \\
&= \sum_{t=1}^{T} -\frac{1}{2} z_t^T [S_t\Psi S_t^T]^{-1} z_t + z_t^T [S_t\Psi S_t^T]^{-1} S_t(C\langle l_t\rangle+\hat{\mu}) - \frac{1}{2}\mathbf{tr}\{[S_t\Psi S_t^T]^{-1} S_t C \langle l_t l_t^T\rangle C^T S_t^T\} \dots \\
&\qquad\qquad -\hat{\mu}^T S_t^T [S_t\Psi S_t^T]^{-1} S_t C\langle l_t\rangle - \frac{1}{2}\hat{\mu}^T S_t^T [S_t\Psi S_t^T]^{-1} S_t \hat{\mu} - \frac{n}{2}\log(2\pi) - \frac{1}{2}\log\left(\mathbf{det}\left\{S_t\Psi_t S_t^T\right\}\right)
\end{aligned}
\tag{C.13}
$$

We now maximize (C.13) with respect to C and Ψ.

Maximization with respect to C. Dropping terms of (C.13) that do not depend on C, we form an objective to maximize

$$
f(C) = \sum_{t=1}^{T} (z_t - S_t\hat{\mu})^T [S_t\Psi S_t^T]^{-1} S_t C \langle l_t\rangle - \frac{1}{2}\mathbf{tr}\{[S_t\Psi S_t^T]^{-1} S_t C \langle l_t l_t^T\rangle C^T S_t^T\} \tag{C.14}
$$

We then calculate

$$
\frac{\partial f}{\partial C(i,:)} = \sum_{t:I_t(i)=1} \frac{\left[(z_t(i) - \hat{\mu}(i))\langle l_t^T\rangle - C(i,:)\langle l_t l_t^T\rangle\right]}{\Psi(i,i)}. \tag{C.15}
$$

Setting (C.15) to zero, we find

$$
\hat{C}(i,;) = \sum_{t:I_t(i)=1} \left([z_t(i) - \hat{\mu}(i)] \langle l_t^T\rangle\right) \left(\sum_{t:I_t(i)=1} \langle l_t l_t^T\rangle\right)^{-1}. \tag{C.16}
$$

Let H_{ij} be the portion of the Hessian of eq. C.14 for $C(i,:)$ and $C(j,:)$. Inspection of eq. C.15 reveals that $H_{ii} = \sum_{t:I_t(i)=1} -\frac{\langle l_t l_t^T\rangle}{\Psi(i,i)}$. By construction $\langle l_t l_t^T\rangle$ is a positive semidefinite matrix, so H_{ii} is negative semidefinite for all i. Inspection also reveals $H_{ij} = 0$ for all $i \neq j$. This establishes that H, the Hessian of eq. C.14 is negative semidefinite. Therefore eq. C.14 is a concave function of C and $\hat{C}$ globally maximizes eq. C.14.

Maximization with Respect to Ψ. Note that Ψ does not appear in eq. C.16 for $\hat{C}$. Therefore, for each M-step and fixed $\hat{\mu}$, eq. C.16 provides a global maximizer of eq. C.12 for all Ψ. Thus, in the derivations that follow we will fix $\hat{\mu}$ and $\hat{C}$ to be those from eqs. C.9 and C.16.

Dropping terms of (C.13) that do not depend on Ψ, we again form an objective to maximize

$$f(\Psi) = \sum_{t=1}^{T} -\frac{1}{2} z_t^T [S_t \Psi S_t^T]^{-1} z_t + z_t^T [S_t \Psi S_t^T]^{-1} S_t (\hat{C} \langle l_t \rangle + \hat{\mu}) - \frac{1}{2} \mathbf{tr}\left\{ [S_t \Psi S_t^T]^{-1} S_t \hat{C} \left\langle l_t l_t^T \right\rangle \hat{C}^T S_t^T \right\} \ldots$$
$$- \hat{\mu}^T S_t^T [S_t \Psi S_t^T]^{-1} S_t \hat{C} \langle l_t \rangle - \frac{1}{2} \hat{\mu}^T S_t^T [S_t \Psi S_t^T]^{-1} S_t \hat{\mu} - \frac{1}{2} \log \left(\mathbf{det} \left\{ S_t \Psi S_t^T \right\} \right) \qquad \text{(C.17)}$$

We derive

$$\frac{\partial f}{\partial \Psi(i,i)} = \sum_{t:I_t(i)=1} \left(\frac{z_t^2(i) - 2 z_t(i)(\hat{C}(i,:) \langle l_t \rangle + \hat{\mu}(i)) + \hat{C}(i,:) \left\langle l_t l_t^T \right\rangle \hat{C}(i,:)^T + 2d(i)\hat{C}(i,;) \langle l_t \rangle + \hat{\mu}^2(i)}{2\Psi^2(i,i)} - \frac{1}{2\Psi(i,i)} \right)$$
$$= \sum_{t:I_t(i)=1} \left(\frac{(z_t(i) - \hat{\mu}(i))^2 - 2(z_t(i) - \hat{\mu}(i))\hat{C}(i,:) \langle l_t \rangle + \hat{C}(i,:) \left\langle l_t l_t^T \right\rangle \hat{C}(i,:)^T}{2\Psi^2(i,i)} - \frac{1}{2\Psi(i,i)} \right) \qquad \text{(C.18)}$$

We now pause to establish one more relation. From eq. C.16, we have
$\hat{C}(i,:) = \sum_{t:I_t(i)=1} \left([z_t(i) - \hat{\mu}(i)] \left\langle l_t^T \right\rangle \right) \left(\sum_{t:I_t(i)=1} \left\langle l_t l_t^T \right\rangle \right)^{-1}$. We can use this to write

$$\hat{C}(i,:) \left(\sum_{t:I_t(i)=1} \left\langle l_t l_t^T \right\rangle \right) \hat{C}(i,:)^T = \sum_{t:I_t(i)=1} [z_t(i) - \hat{\mu}(i)] \, \hat{C}(i,:) \langle l_t \rangle .$$

Using this relation, we can simplify eq. C.18 as

$$\frac{\partial f}{\partial \Psi(i,i)} = \sum_{t:I_t(i)=1} \left(\frac{(z_t(i) - \hat{\mu}(i))^2 - (z_t(i) - \hat{\mu}(i))\hat{C}(i,:) \langle l_t \rangle}{2\Psi^2(i,i)} - \frac{1}{2\Psi(i,i)} \right) . \qquad \text{(C.19)}$$

Setting the left hand side of eq. C.19 to zero, we find that

$$\hat{\Psi}(i,i) = \frac{\sum_{t:I_t(i)=1} (z_t(i) - \hat{\mu}(i))^2 - (z_t(i) - \hat{\mu}(i))\hat{C}(i,:) \langle l_t \rangle}{\sum_{t=1}^{T} I_t(i)} . \qquad \text{(C.20)}$$

Note that $\hat{\Psi}(i,i)$ is unique. We now show that it is a maximizer of eq. C.17. From eq. C.19 it is immediately apparent that $\frac{\partial^2 f}{\partial \Psi(i,i) \partial \Psi(j,j)} = 0$ if $i \neq j$. For $i = j$, we have

$$\frac{\partial^2 f}{\partial \Psi^2(i,i)} = \frac{-\left[\sum_{t:I_t(i)=1} (z_t(i) - \hat{\mu}(i))^2 - (z_t(i) - \hat{\mu}(i))\hat{C}(i,:) \langle l_t \rangle \right]}{\Psi^3(i,i)} + \frac{1}{2\Psi^2(i,i)} . \qquad \text{(C.21)}$$

Evaluating eq. C.21 at $\hat{\Psi}(i,i)$ we find

$$\left. \frac{\partial^2 f}{\partial \Psi^2(i,i)} \right|_{\hat{\Psi}(i,i)} = \frac{-2 \left(\sum_{t=1}^{T} I_t(i) \right)^3 + \left(\sum_{t=1}^{T} I_t(i) \right)^2}{\left[\sum_{t:I_t(i)=1} (z_t(i) - \hat{\mu}(i))^2 - (z_t(i) - \hat{\mu}(i))\hat{C}(i,:) \langle l_t \rangle \right]^2} ,$$

which is always less than or equal to zero. Thus we can conclude that the Hessian of eq. C.17 with respect to the diagonal entries of Ψ is negative semidefinite. Therefore, eq. C.17 is a concave function of the diagonal entries of Ψ and eq. C.20 is a global maximizer of eq. C.17.

C.2 Proofs of Necessity Results

C.2.1 Proof of Theorem 10

Proof. The proof follows the same strategy of the proof for Theorem 9 in the main text. For a K block stitching scenario and an FA model parameterized by $\Theta = \{\mu, C, \Psi\}$, we can write

$$P(z|\Theta) = \prod_{k=1}^{K} \prod_{t \in \eta_k} P(z_t|\Theta), \tag{C.22}$$

where

$$Z_t|\Theta \sim \mathcal{N}\left(\mu(\rho_k), C(\rho_k, :)C(\rho_k, :)^T + \Psi(\rho_k, \rho_k)\right), \qquad k : t \in \nu_k. \tag{C.23}$$

Define $M = CC^T$. Under the assumption that C has full rank submatrices and the assumptions that $|\rho_1| \geq r$, and $|\rho_i \setminus \iota_i| \geq r$ for all $i \geq 2$, it must be that **rank**$\{M\} = r$ and **rank**$\{M(\rho_i \setminus \iota_i, \rho_i \setminus \iota_i) = r\}$ for all $i \geq 2$.

We now consider the problem of reconstructing M from the observed submatrices $M(\rho_1, \rho_1), \ldots, M(\rho_K, \rho_K)$. If $\iota_i < r$ for some $i \geq 2$, Corollary 21 allows us to conclude there exists no unique rank r completion for M.

However, if there is no unique rank r completion for M, this implies there exists a C and C' which are not orthogonal transforms of one another such that $M(\rho_k, \rho_k) = C(\rho_k, :)C(\rho_k, :)^T = C'(\rho_k, :)C'(\rho_k, :)^T$ for all $k \in [K]$. Define $\Theta' = \{\mu, C', \Psi\}$. By eqs. C.24 and C.25, it then follows that $P(z|\Theta) = P(z|\Theta')$. Further, since C and C' are not related through an orthogonal transformation, Θ and Θ' are non-equivalent parameter sets and this completes the proof.

□

C.2.2 Proof of Theorem 11

Proof. The proof follows the same pattern as those for Theorem 9 and Theorem 10. For a K block stitching scenario and an FA model parameterized by $\Theta = \{\mu, C, \Psi\}$, we again write

$$P(z|\Theta) = \prod_{k=1}^{K} \prod_{t \in \eta_k} P(z_t|\Theta), \tag{C.24}$$

where

$$Z_t|\Theta \sim \mathcal{N}\left(\mu(\rho_k), C(\rho_k, :)C(\rho_k, :)^T + \Psi(\rho_k, \rho_k)\right), \qquad k : t \in \nu_k. \tag{C.25}$$

Define $M = CC^T$. Under the assumption that C has full rank submatrices and the assumptions that $|\rho_1| \geq r$, and $|\rho_i \setminus \iota_i| \geq r$ for all $i \geq 2$, it must be that **rank**$\{M\} = r$ and **rank**$\{M(\rho_i \setminus \iota_i, \rho_i \setminus \iota_i) = r\}$ for all $i \geq 2$.

We consider the problem of reconstructing M from the observed submatrices $M(\rho_1, \rho_1), \ldots, M(\rho_K, \rho_K)$. If $\iota_i < r$ for some $i \geq 2$, Corollary 22 allows us to conclude there exists no unique rank r completion for M.

The rest of the proof then follows the same logic as that in the proof of Theorem 10.

□

Appendix D

Appendix for Stitching for Brain Computer Interface

D.1 Learning The Initial Parameters of the Two Stage Decoder

Before the decoder can be used in a self-recalibration mode, it must be initially trained in a supervised manner. For the decoder we use in this study, this occurred on session 0. Once the decoder was trained, its parameters were held fixed across the subsequent days of data we analyze. We now describe how the decoder is initially trained in a supervised manner.

The parameters of the two components of the decoder are learned independently. First, using spike counts vectors collected during a set of trials collected during a training session, the parameters of the FA model are learned. After these parameters are learned the maximum a posteriori estimates of latent states across all bins of the training data are inferred. These estimates of the latent state are then used with estimates of desired velocity to learn the parameters of the Kalman filter.

In a BCI context, determining desired velocity for the purposes of forming a training set of data is difficult as in a clinical setting a subject will not be able to move their limbs. This provides us with no direct observation of the subject's true intent. Lacking this, a common procedure is to train the decoder in an iterative fashion. Previous studies have shown that motor cortical neurons are modulated when a subject is not engaged in but merely observes the performance of a task [101–104]. Thus, we begin by showing the user a series of trials where the cursor is automatically driven along ideal trajectories and we record neural data. We then train the Kalman filter using the displayed trajectories in place of the user's intended trajectories. After this first training step, we allow the user to control the cursor in a shared control mode, where the the cursor's commanded velocities are a mix between ideal velocities that would point straight towards a goal target at all points in time and the velocity that is decoded by the initially trained BCI. After completing a series of trials like this, we again train our decoder, assuming that all points in time, the user desired to move directly towards the target. Training then continues by performing another round of trials under shared control, giving the user more control over the cursor and then retraining the decoder. This process continues until the user is able to reliably control the cursor under full control.

We note that for clarity the BCI decoder was trained like this on day 0 before being used for BCI control on any of the days of data analyzed in the body of the thesis.

Appendix E

Appendix for Deterministic Symmetric Positive Semidefinite Matrix Completion via Nuclear Norm Minimization

E.1 Proof of Lemma 14

Proof. Under conditions $A1$ - $A4$ of chapter 3, $\Omega = \cup_{l=1}^{k} \rho_l \times \rho_l$ for index sets $\rho_1, \ldots, \rho_k$ such that

1. $\cup_{l=1}^{k} \rho_l = [n]$
2. **rank**$\{A(\iota_l, \iota_l)\}$ = r for all $l \geq 2$,

where $\iota_l = \left| \rho_{\tau_l} \cap \left(\cup_{i=1}^{l-1} \rho_{\tau_i} \right) \right|$ for all $l \geq 2$ and some ordering $\tau_1, \ldots, \tau_k$. Without loss of generality assume $\tau_1, \ldots, \tau_k = 1, \ldots, k$, allowing us to define ι_l more simply as $\iota_l = \left| \rho_l \cap \left(\cup_{i=1}^{l-1} \rho_i \right) \right|$.

We now establish the result by induction. Assume that $k = 2$. Then by Lemma 35, we can conclude **rank**$\{A\} = r$. Now assume that $k > 2$. Define $\nu_l = \cup_{i=1}^{l} \rho_i$ and assume **rank**$\{A(\nu_{k-1}, \nu_{k-1})\} = r$. By (1) above it must be that $\rho_k \cup \nu_{k-1} = \cup_{l=1}^{k} \rho_l = [n]$. Further, $\iota_k = \rho_k \cup \nu_{k-1} = |\rho_k \cap \left(\cup_{i=1}^{l-1} \rho_i \right)|$, so from (2), we can conclude that **rank**$\{A(\rho_k \cap \nu_{k-1}, \rho_k \cap \nu_{k-1})\} =$ **rank**$\{A(\iota_k, \iota_k)\} = r$. Finally, since $\iota_k \subseteq \rho_k$ and **rank**$\{A(\iota_k, \iota_k)\} = r$, it must be that **rank**$\{A(\rho_k, \rho_k)\} = r$. We can then use Lemma 35 to conclude **rank**$\{A\} = r$.

□

E.2 Technical Results

We first state a result we will need for certifying the rank of a general matrix when we can only observe the rank of a set of it's submatrices, each submatrix consisting of a set of rows of the original matrix. We will need this to prove Lemma 35 below.

Lemma 34. *Consider the matrix $C \in \mathbb{R}^{n \times m}$. Let $\rho_1, \rho_2 \subseteq [n]$ index the rows of C, and define $\iota = \rho_1 \cap \rho_2$. Assume $\rho_1 \cup \rho_2 = [n]$ and **rank**$\{C(\iota, :)\} =$ **rank**$\{C(\rho_1, :)\} =$ **rank**$\{C(\rho_2, :)\} = r$. Then **rank**$\{C\} = r$.*

Proof. The rank of a matrix, C, is equal to the largest number of linearly independent rows in C. By assumption **rank**$\{C(\iota, :)\} = r$, so $C(\iota, :)$ must contain at least r linearly independent rows. $C(\iota, :)$ is a submatrix of C, so C must also contain at least r linearly independent rows.

We now show C contains no more than r linearly independent rows. By assumption $C(\iota, :)$ can contain no more than r linearly independent rows, as **rank**$\{C(\iota, :)\} = r$. Further, we are unable to find a larger number of linearly independent rows in the entirety of C by considering the additional rows in $C(\rho_1 \setminus \iota, :)$ and $C(\rho_2 \setminus \iota, :)$.

To establish this, we show that every row of $C(\rho_1 \setminus \iota, :)$ and $C(\rho_2 \setminus \iota, :)$ can be formed from a linear combination of r rows in $C(\iota, :)$. We show this for $C(\rho_1 \setminus \iota, :)$ by contradiction. An identical argument applies for $C(\rho_2 \setminus \iota, :)$.

Assume there exists a row in $C(\rho_1 \setminus \iota, :)$ which cannot be formed from a linear combination of rows in $C(\iota, :)$. This then implies this row can be added to a set of r linearly independent rows of $C(\iota, :)$ and the new set of $r + 1$ rows would be linearly independent. However, every row in this new set is a row of the matrix $C(\rho_1, :)$, which by assumption is rank r, and we have achieved a contradiction.

By assumption $\rho_1 \cup \rho_2 = [n]$, so we have shown that C contains at least but no more than r linearly independent rows, and therefore it must be that **rank**$\{C\} = r$.

$\square$

We now use Lemma 34 to establishing sufficient conditions under which we can determine the the rank of a SPSD matrix when we observe two of it's principal submatrices.

Lemma 35. *Consider the matrix $A \in \mathcal{S}_+^n$. Let $\rho_1, \rho_2 \subseteq [n]$ each index rows and columns of A and define $\iota = \rho_1 \cap \rho_2$. Assume $\rho_1 \cup \rho_2 = [n]$ and **rank**$\{A(\iota, \iota)\}$ = **rank**$\{A(\rho_1, \rho_1)\}$ = **rank**$\{A(\rho_2, \rho_2)\} = r$. Then **rank**$\{A\} = r$.*

Proof. In addition to Lemma 34, we will need two intermediate results, which we prove here.

(1) Any matrix $A \in \mathcal{S}_+^n$, can be written as $A = CC^T$ for some $C \in \mathbb{R}^{n \times k}$, where $k =$ **rank**$\{A\}$. To see this let $A = E\Lambda E^T$ be the eigendecomposition of A where $E \in \mathbb{R}^{n \times k}$ is an orthonormal matrix containing the eigenvectors of A and $\Lambda \in \mathbb{R}^{k \times k}$ is a diagonal matrix with the eigenvalues of A along its diagonal. The claim is established by letting $C = E\Lambda^{1/2}$.

(2) For any $A \in \mathbb{R}^{n \times n}$ and $C \in \mathbb{R}^{n \times k}$ such that $A = CC^T$, **rank**$\{A\}$ = **rank**$\{C\}$. This can be shown by letting $C = U\Sigma V^T$ be the singular value decomposition of C, where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{k \times r}$ are orthonormal matrices containing the singular vectors of C and $\Sigma \in \mathbb{R}^{r \times r}$ is a matrix containing the r non-zero singular values of C along it's diagonal. Then $A = CC^T = U\Sigma^2 U^T$ must be rank r as Σ^2 contains r non-zero values along it's diagonal which are the eigenvalues of A.

We now prove the proposition. Consider some $C \in \mathbb{R}^{n \times k}$ such that $A = CC^T$ for the matrix A in the proposition. We have just shown in (1) above that such a C must exist. It must be that $A(\iota, \iota) = C(\iota, :)C(\iota, :)^T$, $A(\rho_1, \rho_1) = C(\rho_1, :)C(\rho_1, :)^T$, $A(\rho_2, \rho_2) = C(\rho_2, :)C(\rho_2, :)^T$. By assumption **rank**$\{A(\iota, \iota)\}$ = **rank**$\{A(\rho_1, \rho_1)\}$ = **rank**$\{A(\rho_2, \rho_2)\}$ = r, so by (2) above, we conclude **rank**$\{C(\iota, :)\}$ = **rank**$\{C(\rho_1, :)\}$ = **rank**$\{C(\rho_2, :)\} = r$. Then by proposition Lemma 34, **rank**$\{C\} = r$. Having shown **rank**$\{C\} = r$, we again use (2) above to establish **rank**$\{A\} = r$. $\square$

Bibliography

[1] György Buzsáki. Large-scale recording of neuronal ensembles. *Nature neuroscience*, 7(5):446–451, 2004. 1

[2] Werner Göbel and Fritjof Helmchen. In vivo calcium imaging of neural network function. *Physiology*, 22(6):358–365, 2007. 1

[3] Leigh R Hochberg, Mijail D Serruya, Gerhard M Friehs, Jon A Mukand, Maryam Saleh, Abraham H Caplan, Almut Branner, David Chen, Richard D Penn, and John P Donoghue. Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature*, 442(7099):164–171, Jul 2006. 1

[4] Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger, and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381:557–564, 2012. 1, 2.1

[5] M.M. Churchland, J.P. Cunningham, M.T. Kaufman, J.D. Foster, P. Nuyujukian, S.I. Ryu, and K.V. Shenoy. Neural population dynamics during reaching. *Nature*, 2012. 1, 1.1.1

[6] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013. 1, 1.1.1

[7] Ofer Mazor and Gilles Laurent. Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons. *Neuron*, 48(4):661–673, 2005. 1, 1.1.1

[8] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014. 1.1.1

[9] Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen I Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. *J Neurophysiol*, 102(1):614–635, Jul 2009. 1.1.1, 1, 6.1.1, 6.2.2

[10] Matthew T Kaufman, Mark M Churchland, Stephen I Ryu, and Krishna V Shenoy. Cortical activity in the null space: permitting preparation without movement. *Nature neuroscience*, 17(3):440–448, 2014. 1.1.1, 4.1, 6.2

[11] K Cora Ames, Stephen I Ryu, and Krishna V Shenoy. Neural dynamics of reaching following incorrect or absent motor preparation. *Neuron*, 81(2):438–451, 2014. 1.1.1

[12] Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, M Yu Byron, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423–426, 2014. 1.1.1, 5.1, 5.2.3, 6.1, 6.1.1, 8.1

[13] Elissa A Hallem and John R Carlson. Coding of odors by a receptor repertoire. *Cell*, 125(1):143–160, 2006. 1.1.1

[14] Marlene R Cohen and John HR Maunsell. When attention wanders: how uncontrolled fluctuations in attention affect performance. *The Journal of Neuroscience*, 31(44):15802–15806, 2011. 1.1.1

[15] Wieland Brendel, Ranulfo Romo, and Christian K Machens. Demixed principal component analysis. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2654–2662. Curran Associates, Inc., 2011. 1.1.1

[16] John O'Keefe and Michael L Recce. Phase relationship between hippocampal place units and the eeg theta

rhythm. *Hippocampus*, 3(3):317–330, 1993. 1.2

[17] Mathew A Wilson and Bruce L McNaughton. Dynamics of the hippocampal ensemble code for space. *Science*, 261:1055–1058, 1993. 1.2

[18] Kelly E Jones, Patrick K Campbell, and Richard A Normann. A glass/silicon composite intracortical electrode array. *Annals of biomedical engineering*, 20(4):423–437, 1992. 1.2

[19] Arnold C Hoogerwerf and Kensall D Wise. A three-dimensional microelectrode array for chronic neural recording. *Biomedical Engineering, IEEE Transactions on*, 41(12):1136–1146, 1994. 1.2

[20] Elizabeth JO Hamel, Benjamin F Grewe, Jones G Parker, and Mark J Schnitzer. Cellular level brain imaging in behaving mammals: An engineering approach. *Neuron*, 86(1):140–159, 2015. 1.2

[21] Philipp J Keller, Annette D Schmidt, Joachim Wittbrodt, and Ernst HK Stelzer. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science*, 322(5904):1065–1069, 2008. 1.2

[22] Srini Turaga, Lars Buesing, Adam M Packer, Henry Dalgleish, Noah Pettit, Michael Hausser, and Jakob Macke. Inferring neural population dynamics from multiple partial recordings of the same neural circuit. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 539–547. Curran Associates, Inc., 2013. 1.2.1, 1.2.2, 4, 8.1

[23] Daniel Soudry, Suraj Keshri, Patrick Stinson, Min-hwan Oh, Garud Iyengar, and Liam Paninski. A shotgun sampling solution for the common input problem in neural connectivity inference. *arXiv preprint arXiv:1309.3724*, 2013. 1.2.1

[24] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *Information Theory, IEEE Transactions on*, 56(6):2980–2998, 2010. 1.2.2, 3.1, 3.1.1

[25] Emmanuel J. Candes and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, May 2010. 1.2.2, 3.1, 3.1.1

[26] Benjamin Recht. A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430, 2011. 1.2.2, 3.1, 3.1.1, 7.1

[27] William E Bishop and Byron M Yu. Deterministic symmetric positive semidefinite matrix completion. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2762–2770. Curran Associates, Inc., 2014. 1.2.2

[28] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. 1.2.2, 3.1.1, 7.1, 7.2

[29] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010. 1.2.2, 3.1, 3.1.1, 3.5, 7.1, 7.2

[30] Emmanuel J Candes and Yaniv Plan. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *Information Theory, IEEE Transactions on*, 57(4):2342–2359, 2011. 1.2.2, 3.1, 3.1.1, 3.5, 7.1

[31] S. Musallam, BD Corneil, B. Greger, H. Scherberger, and RA Andersen. Cognitive control signals for neural prosthetics. *Science*, 305(5681):258–262, 2004. 2.1

[32] Gopal Santhanam, Stephen I Ryu, Byron M Yu, Afsheen Afshar, and Krishna V Shenoy. A high-performance brain-computer interface. *Nature*, 442(7099):195–198, Jul 2006. 2.1, 2.2.3

[33] Meel Velliste, Sagi Perel, M. Chance Spalding, Andrew S Whitford, and Andrew B Schwartz. Cortical control of a prosthetic arm for self-feeding. *Nature*, 453(7198):1098–1101, Jun 2008. 2.1

[34] Zheng Li, Joseph E O'Doherty, Timothy L Hanson, Mikhail A Lebedev, Craig S Henriquez, and Miguel AL Nicolelis. Unscented kalman filter for brain-machine interfaces. *PLoS One*, 4(7):e6243, 2009. 2.1

[35] L.R. Hochberg, D. Bacher, B. Jarosiewicz, N.Y. Masse, J.D. Simeral, J. Vogel, S. Haddadin, J. Liu, S.S. Cash, P. van der Smagt, and J. Donoghue. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012. 2.1

[36] Vikash Gilja, Paul Nuyujukian, Cindy A Chestek, John P Cunningham, Byron M Yu, Joline M Fan, Mark M Churchland, Matthew T Kaufman, Jonathan C Kao, Stephen I Ryu, and Krishna V Shenoy. A high-performance neural prosthesis enabled by control algorithm design. *Nature Neuroscience*, 15:1752–1757, 2012. 2.1, 5.2.3, 5.2.6

[37] Maryam M Shanechi, Rollin C Hu, Marissa Powers, Gregory W Wornell, Emery N Brown, and Ziv M Williams. Neural population partitioning and a concurrent brain-machine interface for sequential motor function. *Nature neuroscience*, 2012. 2.1

[38] UT Eden, W. Truccolo, MR Fellows, JP Donoghue, and EN Brown. Reconstruction of hand movement trajectories from a dynamic ensemble of spiking motor cortical neurons. In *Proc. of the 26th Annual International Conf. of the IEEE EMBS, San Francisco, California*, pages 4017–4020, 2004. 2.1, 5.1

[39] Uri T Eden, Loren M Frank, Riccardo Barbieri, Victor Solo, and Emery N Brown. Dynamic analysis of neural encoding by point process adaptive filtering. *Neural Computation*, 16(5):971–998, May 2004. 2.1, 5.1

[40] L. Srinivasan, U.T. Eden, S.K. Mitter, and E.N. Brown. General-purpose filter design for neural prosthetic devices. *Journal of Neurophysiology*, 98(4):2456–2475, 2007. 2.1, 5.1

[41] Z. Li, J.E. O'Doherty, M.A. Lebedev, and M.A.L. Nicolelis. Adaptive decoding for brain-machine interfaces through bayesian parameter updates. *Neural computation*, 23(11):3162–3204, 2011. 2.1, 2.2.5, 5.1

[42] K. Ohnishi, R.F. Weir, and T.A. Kuiken. Neural machine interfaces for controlling multifunctional powered upper-limb prostheses. *Expert Review of Medical Devices*, 4(1):43–53, 2007. 2.1

[43] R. Scherer, GR Muller, C. Neuper, B. Graimann, and G. Pfurtscheller. An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate. *IEEE Transactions on Biomedical Engineering*, 51(6):979–984, 2004. 2.1

[44] P. Brunner, A.L. Ritaccio, J.F. Emrich, H. Bischof, and G. Schalk. Rapid communication with a "p300" matrix speller using electrocorticographic signals (ecog). *Frontiers in Neuroscience*, 5:1–9, 2011. 2.1

[45] Tobias Pistohl, Andreas Schulze-Bonhage, Ad Aertsen, Carsten Mehring, and Tonio Ball. Decoding natural grasp types from human ecog. *Neuroimage*, 59(1):248–260, 2012. 2.1

[46] Cynthia A Chestek, Vikash Gilja, Christine H Blabe, Brett L Foster, Krishna V Shenoy, Josef Parvizi, and Jaimie M Henderson. Hand posture classification using electrocorticography signals in the gamma band over human sensorimotor brain areas. *Journal of Neural Engineering*, 10(2):026002, 2013. 2.1

[47] V. Aggarwal, S. Acharya, F. Tenore, Hyun-Chool Shin, R. Etienne-Cummings, M. H. Schieber, and N. V. Thakor. Asynchronous decoding of dexterous finger movements using m1 neurons. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 16(1):3–14, 2008. 2.1

[48] W. Wang, A. D. Degenhart, J. L. Collinger, R. Vinjamuri, G. P. Sudre, P.D. Adelson, D. L. Holder, E.C. Leuthardt, D.W. Moran, M. L. Boninger, A.B. Schwartz, D.J. Crammond, E. C. Tyler-Kabara, and D.J. Weber. Human motor cortical activity recorded with micro-ecog electrodes, during individual finger movements. In *Proc. of the 31st Annual International Conf. of the IEEE EMBS, Minneapolis, MN*, pages 586–589, 2009. 2.1

[49] Krishna V Shenoy, Daniella Meeker, Shiyan Cao, Sohaib A Kureshi, Bijan Pesaran, Christopher A Buneo, Aaron P Batista, Partha P Mitra, Joel W Burdick, and Richard A Andersen. Neural prosthetic control signals from plan activity. *Neuroreport*, 14(4):591–596, 2003. 2.1

[50] Neil Achtman, Afsheen Afshar, Gopal Santhanam, M Yu Byron, Stephen I Ryu, and Krishna V Shenoy. Free-paced high-performance brain–computer interfaces. *Journal of Neural Engineering*, 4:336–347, 2007. 2.1, 2.4

[51] C. Kemere, G. Santhanam, Byron M. Yu, A. Afshar, S.I. Ryu, T.H. Meng, and K.V. Shenoy. Detecting neural-state transitions using hidden markov models for motor cortical prostheses. *Journal of Neurophysiology*, 100(4):2441–2452, 2008. 2.1, 2.4

[52] Z. Wang, A. Gunduz, P. Brunner, A.L. Ritaccio, Q. Ji, and G. Schalk. Decoding onset and direction of

movements using electrocorticographic (ecog) signals in humans. *Frontiers in Neuroengineering*, 5:15, 2012. 2.1

[53] Robert Leeb, Doron Friedman, Gernot R Müller-Putz, Reinhold Scherer, Mel Slater, and Gert Pfurtscheller. Self-paced (asynchronous) bci control of a wheelchair in virtual environments: a case study with a tetraplegic. *Computational intelligence and neuroscience*, 2007, 2007. 2.1

[54] Ferran Galán, Marnix Nuttin, Eileen Lew, Pierre W Ferrez, Gerolf Vanacker, Johan Philips, and J del R Millán. A brain-actuated wheelchair: asynchronous and non-invasive brain–computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, 2008. 2.1

[55] P. Shenoy, M. Krauledat, B. Blankertz, R.P.N. Rao, and K.R. Müller. Towards adaptive classification for bci. *Journal of Neural Engineering*, 3:R13–R23, 2006. 2.1

[56] J Pascual, C Vidaurre, and M Kawanabe. Investigating eeg non-stationarities with robust pca and its application to improve bci performance. *International Journal of Bioelectromagnetism*, 13(1):50–51, 2011. 2.1

[57] P. Sykacek and S. Roberts. Adaptive classification by variational kalman filtering. In S. Thrun, S. Becker, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, pages 737–744, Cambridge, MA, 2002. MIT Press. 2.1

[58] P. Sykacek, S.J. Roberts, and M. Stokes. Adaptive bci based on variational bayesian kalman filtering: an empirical evaluation. *IEEE Transactions on Biomedical Engineering*, 51(5):719–727, 2004. 2.1

[59] J. Blumberg, J. Rickert, S. Waldert, A. Schulze-Bonhage, A. Aertsen, and C. Mehring. Adaptive classification for brain computer interfaces. In *Proc. of the 29th Annual International Conf. of the IEEE EMBS, Lyon, France*, pages 2536–2539, 2007. 2.1

[60] C. Vidaurre, A. Schlögl, B. Blankertz, M. Kawanabe, and K.R. Müller. Unsupervised adaptation of the lda classifier for brain-computer interfaces. In *Proceedings of the 4th International Brain-Computer Interface Workshop and Training Course, Graz, Austria*, volume 2008, pages 122–127, 2008. 2.1

[61] C. Vidaurre and B. Blankertz. Towards a cure for bci illiteracy. *Brain Topography*, 23(2):194–198, 2010. 2.1

[62] G. Santhanam, B.M. Yu, V. Gilja, S.I. Ryu, A. Afshar, M. Sahani, and K.V. Shenoy. Factor-analysis methods for higher-performance neural prostheses. *Journal of Neurophysiology*, 102(2):1315–1330, 2009. 2.1

[63] G.W. Fraser, S.M. Chase, A. Whitford, and A.B. Schwartz. Control of a brain–computer interface without spike sorting. *Journal of Neural Engineering*, 6:055004, 2009. 2.1, 2.2

[64] C.A. Chestek, V. Gilja, P. Nuyujukian, J.D. Foster, J.M. Fan, M.T. Kaufman, M.M. Churchland, Z. Rivera-Alvidrez, J.P. Cunningham, S.I. Ryu, et al. Long-term stability of neural prosthetic control signals from silicon cortical arrays in rhesus macaque motor cortex. *Journal of Neural Engineering*, 8(4):045005, 2011. 2.1, 2.4

[65] Cindy A Chestek, John P Cunningham, Vikash Gilja, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural prosthetic systems: current problems and future directions. In *Proc. of the 31st Annual International Conf. of the IEEE EMBS, Minneapolis, MN*, pages 3369–3375, 2009. 2.2

[66] G. Santhanam, M.D. Linderman, V. Gilja, A. Afshar, S.I. Ryu, T. Meng, and K. Shenoy. HermesB: a continuous neural recording system for freely behaving primates. *IEEE Transactions on Biomedical Engineering*, 54(11):2037–2050, 2007. 2.2.2, 2.4

[67] V.S. Polikov, P.A. Tresco, and W.M. Reichert. Response of brain tissue to chronically implanted neural electrodes. *Journal of Neuroscience Methods*, 148(1):1–18, 2005. 2.2.2

[68] C.A. Chestek, A.P. Batista, G. Santhanam, B.M. Yu, A. Afshar, J.P. Cunningham, V. Gilja, S.I. Ryu, M.M. Churchland, and K.V. Shenoy. Single-neuron stability during repeated reaching in macaque premotor cortex. *Journal of Neuroscience*, 27(40):10742–10750, 2007. 2.2.2

[69] Dawn M Taylor, Stephen I Helms Tillery, and Andrew B Schwartz. Direct cortical control of 3d neuroprosthetic devices. *Science*, 296(5574):1829–1832, 2002. 2.2.2

[70] Beata Jarosiewicz, Steven M Chase, George W Fraser, Meel Velliste, Robert E Kass, and Andrew B Schwartz. Functional network reorganization during learning in a brain-computer interface paradigm. *Proceedings of the National Academy of Sciences*, 105(49):19486–19491, 2008. 2.2.2

[71] Karunesh Ganguly and Jose M Carmena. Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biology*, 7(7):e1000153, Jul 2009. 2.2.2, 6.1

[72] Steven M Chase, Robert E Kass, and Andrew B Schwartz. Behavioral and neural correlates of visuomotor adaptation observed through a brain-computer interface in primary motor cortex. *Journal of Neurophysiology*, 108(2):624–644, 2012. 2.2.2

[73] EM Maynard, NG Hatsopoulos, CL Ojakangas, BD Acuna, JN Sanes, RA Normann, and JP Donoghue. Neuronal interactions improve cortical population coding of movement direction. *The Journal of Neuroscience*, 19(18):8083–8093, 1999. 2.2.3

[74] H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *Automatic Control, IEEE Transactions on*, 33(8):780–783, 1988. 2.2.4

[75] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 2.2.4, 4.4, C.1.3

[76] Steven M Chase, Andrew B Schwartz, and Robert E Kass. Bias, optimal linear estimation, and the differences between open-loop simulation and closed-loop performance of spiking-based brain–computer interface algorithms. *Neural Networks*, 22(9):1203–1213, 2009. 2.3.3

[77] Teresa M Vaughan, Dennis J McFarland, Gerwin Schalk, William A Sarnacki, Dean J Krusienski, Eric W Sellers, and Jonathan R Wolpaw. The wadsworth bci research and development program: at home with bci. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):229–233, 2006. 2.4

[78] J'anos A Perge, Mark L Homer, Wasim Q Malik, Sydney Cash, Emad Eskandar, Gerhard Friehs, John P Donoghue, and Leigh R Hochberg. Intra-day signal instabilities affect decoding performance in an intracortical neural interface system. *Journal of Neural Engineering*, 10(3):036004, 2013. 2.4

[79] Robert D Flint, Zachary A Wright, Michael R Scheid, and Marc W Slutzky. Long term, stable brain machine interface performance using local field potentials and multiunit spikes. *Journal of neural engineering*, 10(5):056005, 2013. 2.4

[80] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 3.1

[81] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, Cambridge, MA, 2006. 3.1, C.1.3

[82] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007. 3.1

[83] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11(2057-2078):1, 2010. 3.1, 3.1.1, 3.5

[84] Vladimir Koltchinskii, Karim Lounici, and Alexandre B Tsybakov. Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39(5):2302–2329, 2011. 3.1, 3.1.1, 3.5

[85] Sahand Negahban and Martin J Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *The Journal of Machine Learning Research*, 13:1665–1697, 2012. 3.1, 3.1.1

[86] Akshay Krishnamurthy and Aarti Singh. Low-rank matrix and tensor completion via adaptive sampling. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 836–844. 2013. 3.1, 3.1.1, 7.1

[87] Jie Chen, Nannan Cao, Kian Hsiang Low, Ruofei Ouyang, Colin Keng-Yan Tan, and Patrick Jaillet. Parallel gaussian process regression with low-rank covariance matrix approximations. *arXiv preprint arXiv:1305.5826*, 2013. 3.1, 3.1.1, 7.1

[88] Eyal Heiman, Gideon Schechtman, and Adi Shraibman. Deterministic algorithms for matrix completion.

Random Structures & Algorithms, 2013. 3.1.1, 7.1

[89] Troy Lee and Adi Shraibman. Matrix completion from any given set of observations. In *Advances in Neural Information Processing Systems*, pages 1781–1787, 2013. 3.1.1

[90] Monique Laurent. Matrix completion problems. *Encyclopedia of Optimization*, pages 1967–1975, 2009. 3.1.1

[91] Monique Laurent and Antonios Varvitsiotis. A new graph parameter related to bounded rank positive semidefinite matrix completions. *Mathematical Programming*, 145(1-2):291–325, 2014. 3.1.1

[92] Monique Laurent and Antonios Varvitsiotis. Positive semidefinite matrix completion, universal rigidity and the strong arnold property. *Linear Algebra and its Applications*, 452:292–317, 2014. 3.1.1

[93] Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001. 3.1.1

[94] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nystrom method. In *International Conference on Artificial Intelligence and Statistics*, pages 304–311, 2009. 3.1.1

[95] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966. 3.3, B.2, B.2

[96] Joao Semedo, Amin Zandvakili, Adam Kohn, Christian K Machens, and Byron M Yu. Extracting latent structure from multiple interacting neural populations. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2942–2950. Curran Associates, Inc., 2014. 4.1, 6.2

[97] Charles Spearman. "General intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904. 4.2

[98] B.S. Everitt. *An Introduction to Latent Variable Models (Monographs on Statistics and Applied Probability*. Chapman and Hall, 1984. 4.2, C.1.1

[99] Alexander T Basilevsky. *Statistical factor analysis and related methods: theory and applications*, volume 418. John Wiley & Sons, 2009. 4.2

[100] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. 5.1

[101] Paul Cisek and John F Kalaska. Neural correlates of mental rehearsal in dorsal premotor cortex. *Nature*, 431(7011):993–996, 2004. 5.2.2, D.1

[102] Remy Wahnoun, Jiping He, and Stephen I Helms Tillery. Selection and parameterization of cortical neurons for neuroprosthetic control. *Journal of neural engineering*, 3(2):162, 2006. 5.2.2, D.1

[103] Dennis Tkach, Jacob Reimer, and Nicholas G Hatsopoulos. Congruent activity during action and action observation in motor cortex. *The Journal of Neuroscience*, 27(48):13241–13250, 2007. 5.2.2, D.1

[104] Dennis Tkach, Jake Reimer, and Nicholas G Hatsopoulos. Observation-based learning for brain–machine interfaces. *Current opinion in neurobiology*, 18(6):589–594, 2008. 5.2.2, D.1

[105] Wei Wu, Yun Gao, Elie Bienenstock, John P Donoghue, and Michael J Black. Bayesian population decoding of motor cortical activity using a kalman filter. *Neural Computation*, 18(1):80–118, 2006. 5.2.3

[106] Sung-Phil Kim, John D Simeral, Leigh R Hochberg, John P Donoghue, and Michael J Black. Neural control of computer cursor velocity by decoding motor cortical spiking activity in humans with tetraplegia. *Journal of neural engineering*, 5(4):455, 2008. 5.2.3

[107] Rony Paz, Chen Natan, Thomas Boraud, Hagai Bergman, and Eilon Vaadia. Emerging patterns of neuronal responses in supplementary and primary motor areas during sensorimotor adaptation. *The Journal of neuroscience*, 25(47):10941–10951, 2005. 6.1

[108] Daniel Durstewitz, Nicole M Vittoz, Stan B Floresco, and Jeremy K Seamans. Abrupt transitions between prefrontal neural ensemble states accompany behavioral transitions during rule learning. *Neuron*, 66(3):438–448, 2010. 6.1

[109] Adam S Dickey, Aaron Suminski, Yali Amit, and Nicholas G Hatsopoulos. Single-unit stability using chronically implanted multielectrode arrays. *Journal of neurophysiology*, 102(2):1331–1339, 2009. 6.1.1

[110] George W Fraser and Andrew B Schwartz. Recording from the same neurons chronically in motor cortex. *Journal of neurophysiology*, 107(7):1970–1978, 2012. 6.1.1

[111] Christian Büchel and KJ Friston. Modulation of connectivity in visual pathways by attention: cortical interactions evaluated with structural equation modelling and fmri. *Cerebral cortex*, 7(8):768–778, 1997. 6.2

[112] James Rowe, Klaas Enno Stephan, Karl Friston, Richard Frackowiak, Andrew Lees, and Richard Passingham. Attention to action in parkinson's disease. *Brain*, 125(2):276–289, 2002. 6.2

[113] Daniel Baldauf and Robert Desimone. Neural mechanisms of object-based attention. *Science*, 344(6182):424–427, 2014. 6.2

[114] Eugenio Rodriguez, Nathalie George, Jean-Philippe Lachaux, Jacques Martinerie, Bernard Renault, and Francisco J Varela. Perception's shadow: long-distance synchronization of human brain activity. *Nature*, 397(6718):430–433, 1999. 6.2

[115] J Sarnthein, Hellmuth Petsche, P Rappelsberger, GL Shaw, and A Von Stein. Synchronization between prefrontal and posterior association cortex during human working memory. *Proceedings of the National Academy of Sciences*, 95(12):7092–7096, 1998. 6.2

[116] Peter Konigqt and Wolf Singer. Visuomotor integration is associated with zero time-lag synchronization among cortical areas. *Nature*, 385:157, 1997. 6.2

[117] Corrado Bernasconi, Astrid von Stein, Carl Chiang, and Peter KoÈnig. Bi-directional interactions between visual areas in the awake behaving cat. *Neuroreport*, 11(4):689–692, 2000. 6.2

[118] Georgia G Gregoriou, Stephen J Gotts, Huihui Zhou, and Robert Desimone. High-frequency, long-range coupling between prefrontal and visual cortex during attention. *Science*, 324(5931):1207–1210, 2009. 6.2

[119] Georgia G Gregoriou, Stephen J Gotts, and Robert Desimone. Cell-type-specific synchronization of neural activity in fef with v4 during attention. *Neuron*, 73(3):581–594, 2012. 6.2

[120] Alan Julian Izenman. Reduced-rank regression for the multivariate linear model. *Journal of multivariate analysis*, 5(2):248–264, 1975. 6.2.1, 6.2.1

[121] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004. 6.2.1

[122] Amin Zandvakili and Adam Kohn. Coordinated neuronal activity enhances corticocortical communication. *Neuron*, 87(4):827–839, 2015. 6.2.1

[123] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *Learning Theory*, pages 545–560. Springer, 2005. 7.1

[124] Stefan Fischer, Manfred Hallschmid, Anna Lisa Elsner, and Jan Born. Sleep forms memory for finger skills. *Proceedings of the National Academy of Sciences*, 99(18):11987–11991, 2002. 8.2

[125] Matthew P Walker, Tiffany Brakefield, Joshua Seidman, Alexandra Morgan, J Allan Hobson, and Robert Stickgold. Sleep and the time course of motor skill learning. *Learning & Memory*, 10(4):275–284, 2003. 8.2

[126] Gilbert W Stewart and Ji-guang Sun. *Matrix perturbation theory*. Academic press, 1990. B.2, B.2

[127] Ren-Cang Li. New perturbation bounds for the unitary polar factor. *SIAM Journal on Matrix Analysis and Applications*, 16(1):327–332, 1995. B.2

[128] Jan R Magnus and Heinz Neudecker. Matrix differential calculus with applications in statistics and econometrics. 1999. C.1.3