Semi-Supervised Classification for Intracortical Brain-Computer Interfaces

William Bishop wbishop@cs.cmu.edu
Department of Machine Learning
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract

Intracortical brain-computer interface (BCI) systems may one day allow paralyzed patients to interface with robotic arms or computer programs using their thoughts alone. However, a common and unaddressed issue with these systems is that due to small instabilities in the recorded signals, the decoding algorithms they rely upon must be retrained daily in a supervised manner. While this may be acceptable in a laboratory, it presents a burden to the patient. In this work, using data recorded from two subjects over a period of 41 (36) separate days, we first investigate the behavior of a standard decoder without daily retraining. Contrary to what might be expected, we find that mean daily accuracy does not significantly decline with time, though daily performance may be highly variable without retraining. Second, we investigate the behavior of the neural signals in both datasets across time and find that within day changes of the signal characteristics are largely dwarfed by between day changes. Finally, we present two algorithms which adapt to changes in neural signals in a semi-supervised manner and demonstrate stable decoding on both datasets without any daily supervised retraining.

Keywords: Intracortical Brain-Computer Interface, Semi-Supervised Learning, Adaptive Classifiers

1. Introduction

Brain-computer interfaces (BCI) aim to assist disabled patients by translating recorded neural activity into control signals for artificial devices. Intracortical BCI systems rely upon microelectrode arrays chronically implanted in the cortex to record the activity of populations of tens to hundreds of neurons. Such systems can be grouped into two categories: those that decode a continuously valued control signal, such as the position of a computer cursor (Hochberg et al. (2012)) or prosthetic limb (Velliste et al. (2008)) and those that classify a discrete control signal, such as the final, intended target of a movement (Santhanam et al. (2006); Musallam et al. (2004)).

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 the online, closed-loop setting. In almost all cases these decoders must be retrained daily in a supervised manner to maintain their accuracy (e.g., Hochberg et al. (2012)). 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 and so labelled data for training can be collected. However, it is not clear that the burden of such retraining will be feasible in a clinical setting (Gilja et al. (2011)).

Adaptive decoders, which simultaneously estimate decode parameters along with the user's intent, do not require supervised retraining. Previous work on adaptive decoders for intracortical BCI has focused on decoding continuous actions (Eden et al. (2004); Li et al. (2011); Srinivasan et al. (2007); Nuyujukian et al. (2012)). While continuous decoders have many important applications, classifiers of discrete actions may be better suited for some BCI applications, such as interfacing with a menu or a spelling application. Indeed, Santhanam et al. (Santhanam et al. (2006)) 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 a discrete selection.

In this paper, we undertake an investigation of intracortical BCI classifier instability. We do so using datasets collected from two subjects performing a center-out reaching task on 41 (36) separate days. We describe these datasets in Section 2, and then examine the performance of a standard BCI decoder with retraining on both datasets in Section 3. We show that contrary to what might be expected, accuracy of the standard decoder does not decline to chance as time from training increases, though performance on any given day may be highly unpredictable.

We then seek to develop novel classifiers which have consistently high performance from day-to-day without retraining. To do this, we first investigate the relative magnitude of changes in the parameters that characterized neural signals (referred to as a "tuning-parameters") between and within days in Section 4. We find that the scale of between-day changes in tuning parameters largely dwarfs within-day changes. We use this insight to propose two generative models for BCI signals in Section 5 and present recursive Bayesian methods for decoding brain signals generated according to these models in Section 6. In Section 7, we present a method of making the initially proposed decoding techniques more robust for real-world use. In Section 8, we describe methods for learning the model parameters and finally, in Section 9, we present results demonstrating stable decoding without any daily retraining on the same two datasets analyzed throughout this paper.

2. Two Prerecorded Datasets for the Design of Stable Intracortical BCI Algorithms

The datasets analyzed in this study consist of neural recordings from two Rhesus macaque monkeys, referred to as monkey L and monkey I. Animal protocols were approved by the Stanford Institutional Animal Care and Use Committee.

Before describing the experiment, we pause briefly to describe the neural signals that were recorded in these experiments. Cells in the brain, referred to as neurons, encode information through the generation of a time series of distinct voltage "spikes" referred to as action potentials. Electrodes implanted in the brain of a trained animal can record these voltage spikes, and a crude but computationally efficient means of detecting them is to simply set upper and lower threshold levels (Chestek et al. (2009); Fraser et al. (2009)) and count a spike anytime the voltage level on an electrode crosses one of these.

In the present work, microelectrode arrays with 96 recording electrodes were implanted in the motor cortex of both animals and voltage threshold crossings were recorded while the animals performed a series of trials on each day. Fig. 1 illustrates the experimental setup. Targets were presented visually on a screen in front of the monkey. A single trial began with the monkey placing his hand on a central target for a pseudo-randomly chosen hold time. After the hold period was finished, a go cue was issued in the form of the appearance of a peripheral target and the animal was allowed to immediately reach for the target. After holding the peripheral target for another pseudo-randomly generated hold time, the animal was administered a liquid reward.

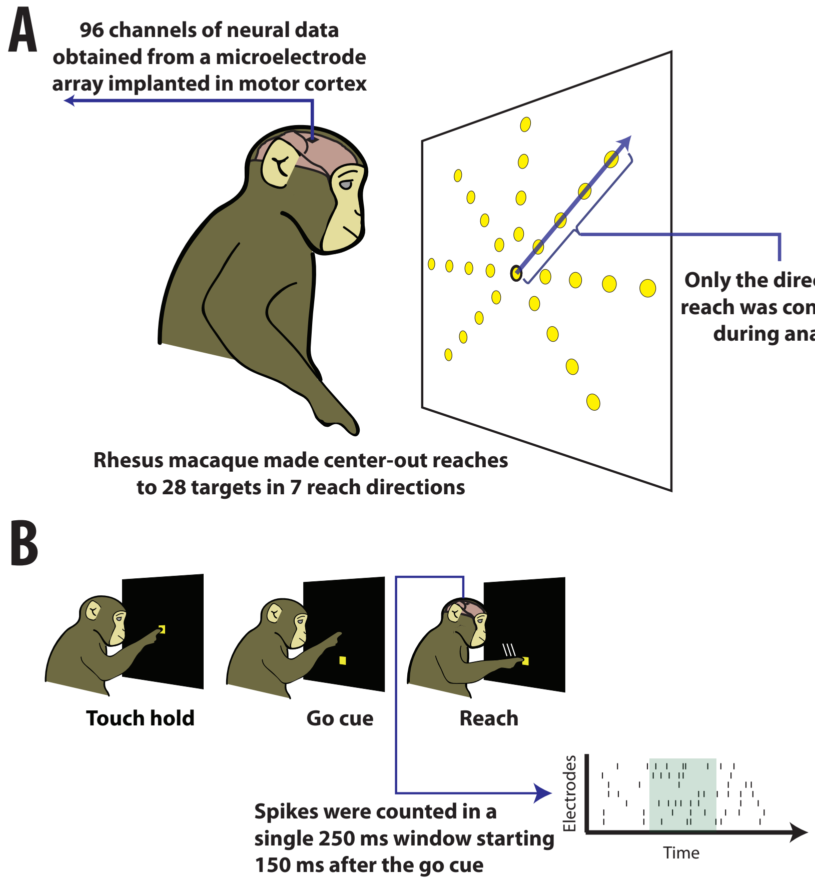

A
96 channels of neural data obtained from a microelectrode array implanted in motor cortex
Only the direction of reach was considered during analysis
Rhesus macaque made center-out reaches to 28 targets in 7 reach directions
B
Touch hold
Go cue
Reach
Spikes were counted in a single 250 ms window starting 150 ms after the go cue
Electrodes
Time

Figure 1: The experimental paradigm used during the collection of the two datasets analyzed in this paper. A) On each trial, the animal made a reach to one of 28 peripheral targets while neural signals were recorded from a microlectrode array implanted in the animal's motor cortex. Only reach direction, and not target distance from center, was considered when labeling trials for later analysis. B) A trial consisted of a central hold period, followed by a reach and a hold at the final target. Feature vectors for each trial were formed by counting spikes in a single 250 millisecond window during each reach.

Peripheral 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 the rest of this paper, we will collapse across target distance, and label trials using only the direction of reach.

The goal of this work is to decode the direction of reach from neural data. To do this we must form a feature vector from the neural spike data recorded for each trial. A central tenet of neuroscience is that in many parts of the brain it is not the exact timing of action potentials but their rate of generation that carries information. Therefore, we form a feature vector for each trial by simply counting the number of threshold crossings (referred to as spike counts in this paper) on each electrode of the recording array in a single 250 millisecond window starting 150 milliseconds after the go cue. The 150 ms lag is chosen to account for the time it takes for the neural signals in motor cortex to reflect the desired reach movement.

Throughout this paper, when stating results for both monkeys, results for monkey L will be stated first, followed by results for monkey I in parentheses. Recordings were made on a total of 41 (36) days spanning a period of 48 (58) days in total. Only successful trials were considered in this analysis. There were (mean $\pm$ s.d.) 2510 ± 661 (1783 ± 426) successful trials spanning 78 ± 18 (91 ± 16) minutes per day.

Before moving on, we pause to address one point that may have left the reader new to the BCI literature confused. Specifically, while this is study is aimed at developing new decoders for BCI use, the monkeys are not performing a BCI task. They are selecting targets with their native limbs. While all BCI algorithms must eventually be validated in a closed loop setting, it is common in the field to develop and test initial algorithms in an offline setting with prerecorded data where the goal is to predict natural movement from brain signals. As this work is the first (to our knowledge) to undertake the development of stable classifiers for intracortical BCI, we choose to analyze prerecorded datasets and save online verification for a later study.

3. Performance of a Standard BCI Classifier without Retraining

We now explore the performance of a standard decoder which is not retrained daily. Let $y_{d,t} \in \{1, \ldots, 7\}$ indicate the reach direction for trial t on day d and $\boldsymbol{x}_{d,t}$ indicate the spike counts for all electrodes during trial t on day d. If $x_{d,e,t}$ denotes element e of the vector $\boldsymbol{x}_{d,t}$, and $\lambda_{d,e,j}$ indicates the expected number of spike counts on electrode e when the monkey reaches in direction j on day d, a common decoding model is

$$y_{d,t} \sim \text{Uniform}\left(\frac{1}{7}\right) \tag{1}$$

$$P(x_{d,e,t} | y_{d,t} = j) = \text{Poisson}(\lambda_{d,e,j}) \tag{2}$$

$$P(\boldsymbol{x}_{d,t} | y_{d,t} = j) = \prod_{e=1}^{E} P(x_{d,e,t} | y_{d,t} = j), \tag{3}$$

which is a generative model for a naïve Bayes classifier with a Poisson observation model. This type of model has been used successfully to decode discrete BCI data in the past (Baker et al. (2009), Santhanam et al. (2006)).

It is typical practice to train and test this classifier on data collected on the same day, and we consider the performance of this retrained classifier as baseline. We use the first 400 trials on each day to learn the maximum likelihood (MLE) estimates for the $\lambda_{d,e,j}$ parameters and then estimate $\hat{y}_{t,d}$ for trials 401 and on as

$$\hat{y}_{d,t} = \underset{j}{\operatorname{argmax}} P(y_{d,t} = j | \boldsymbol{x}_{d,t}), \tag{4}$$

where $P(y_{d,t} = j|\boldsymbol{x}_{d,t})$ can be obtained from Bayes' rule and the equations defined immediately above. We would also like to examine the performance of this decoder when it is not retrained daily, which we refer to as the static decoder. To do this, we define $\lambda_{d,e,j}$ to be constant across days and learn the MLE estimates for these parameters from the first ten days of data for each dataset. Having fixed the $\lambda_{d,e,j}$ values, we then predict reach direction, $\hat{y}_{d,t}$, on trials 401 and on of the remaining 31 (26) days for each dataset. Note that when learning the $\lambda_{d,e,j}$ values, we remove any electrodes from the decoder which have marginal firing rates of less than 2 counts per trial over all trials in the first 10 days of training data to improve decoder reliability. This results in the removal of 4 electrodes for monkey L and 0 electrodes for monkey I.

The results of this analysis are shown in Fig. 2. As expected, retraining of the classifier results in consistently high mean daily accuracy. The careful reader will notice a precipitous drop in accuracy of the retrained decoder for monkey I on day 23. Analysis of the data indicates a sharp change in the average number of threshold crossings across many of the electrodes during this day around trial 1140. This was likely caused by an experimenter manually reseting the threshold levels of the spike detection algorithm when the original data was being recorded, rendering the $\lambda_{d,e,j}$ values that were learned on trials 1-400 obsolete and explaining the sharp drop in performance.

Performance of the static decoder is highly variable from day to day but it is noteworthy that the expected performance of the decoder does not seem to decline with time. Indeed, the 95% confidence interval for the slope of a linear fit of mean daily performance for monkey M is (-0.005 %/day, 0.005 %/day) and (-0.012573 %/day, 0.001665 %/day) for monkey I, indicating the slope is not significantly different than zero in either case.

This result is noteworthy for two reasons. First, there is little previously published work on the behavior of standard algorithms without retraining. Without experimental evidence, it would not be unreasonable to assume that signals would randomly walk from day-to-day and the decode accuracy of a static decoder would drop to chance. Based on this analysis from two different monkeys, this does not appear to be the case. Second, while performance does not drop to chance it is highly variable from day-to-day. This variability would still present a major problem in the clinic, and we now turn our attention to designing algorithms which will address this. We begin in the next section by a characterization of how the underlying neural signals change between and within days.

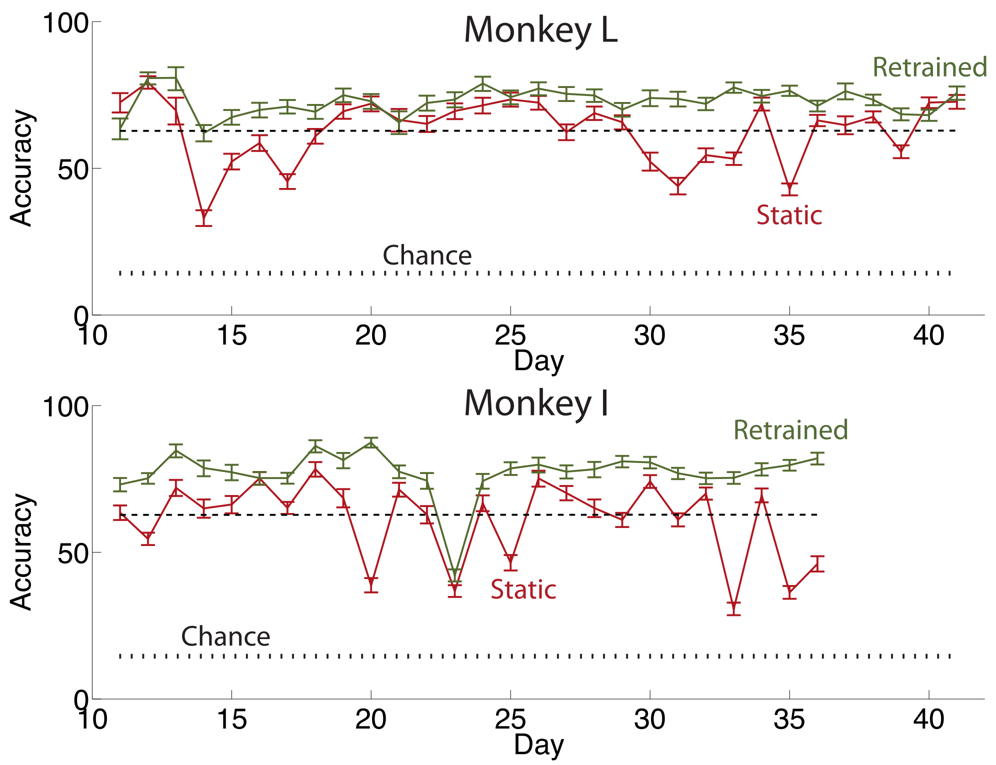


Figure 2: The mean daily accuracy with 95% score confidence intervals (Agresti and Coull (1998))of the naïve Bayes Poisson classifier on trials 401 and on for days 11 and on for monkeys L and I when retrained (green) on the first 400 trials of each day and when decode parameters are held static (red) after being learned from days 1-10. The black dashed line shows a linear fit to mean daily performance for the static decoders in both datasets. The slope of this fit is not significantly different from zero for either monkey.

4. Tuning Parameter Drift Between and Within Days

We now turn to a brief examination of how neural signals change within and between days. To do this, we must first introduce the concept of a tuning parameter, which we define as the expected number of spike counts on electrode e conditioned on the monkey reaching in direction j on day d. This corresponds to our definition of $\lambda_{d,e,j}$ in Section 3 above and without distinction we will to refer to these $\lambda_{d,e,j}$ values as "decode parameters" or "tuning parameters." While the exact mechanisms remain unknown, movements of the recording array relative to the brain (Santhanam et al. (2007)), scar tissue buildup (Polikov et al. (2005)), behavioral changes (Chestek et al. (2007)) and neural plasticity (Ganguly and Carmena (2009)) may all cause these tuning parameters to drift.

In this analysis we concatenate the trials for each day and perform moving averages of spike counts for each trial with a Gaussian kernel across all trials for the same reach direction for each electrode to form smoothed estimates of the underlying tuning parameters through time. The results shown in Fig. 3 A and B were obtained with a kernel with a standard deviation of 100 trials, and results obtained with kernels with widths of 25, 50 and 200 trials were qualitatively similar.

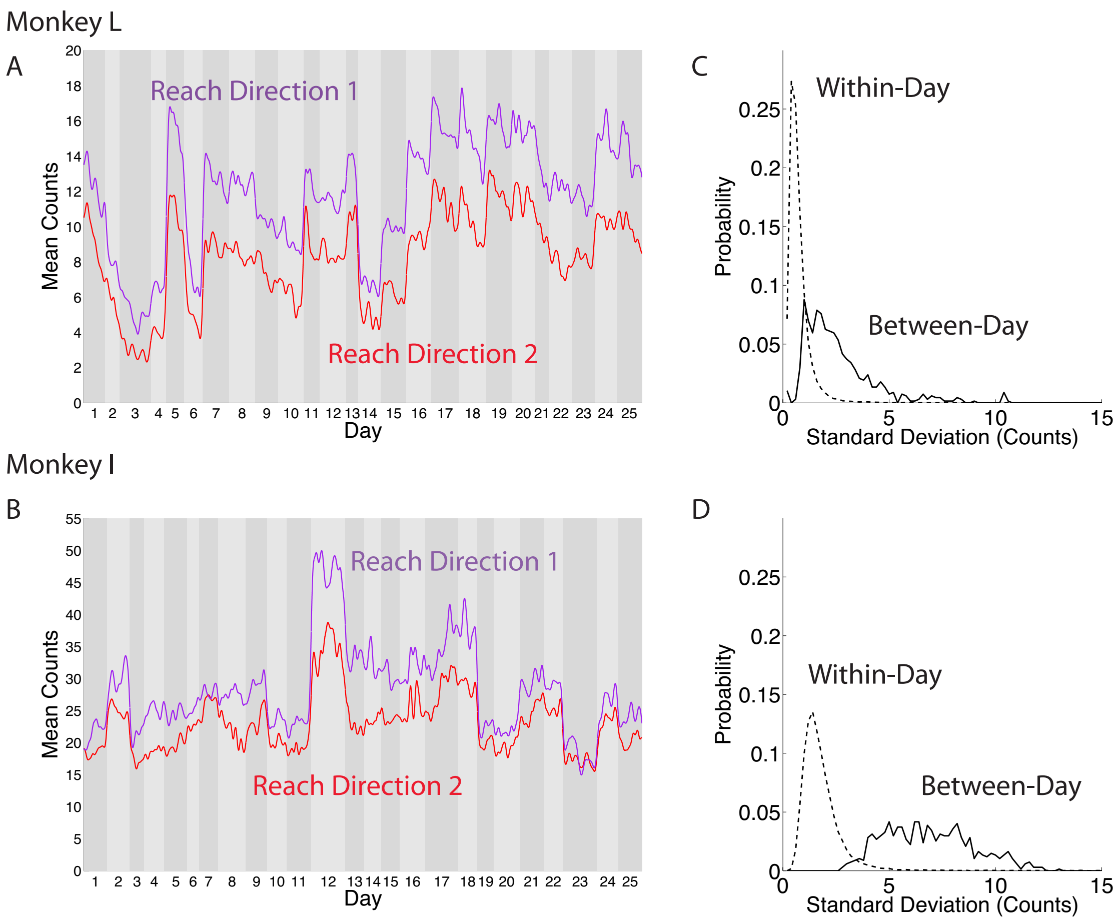


Figure 3: A, B: Estimates of tuning parameters obtained by smoothing spike counts with a Gaussian kernel with a standard deviation of 100 trials for monkeys M and I. Estimated parameters for two reach directions for the same electrode for the first 25 days of data are shown for each monkey. Separate days are indicated by the shaded stripes. C, D: Histograms of the standard deviation of tuning parameter values calculated within days (dashed line) and between days (solid line). The x-axis of this panel has been clipped for panel C so that .08% (0%) of the probability mass lies outside of the figure to the right for the within (between)-day plots. Similarly, the x-axis has been clipped for panel D so that .24% (0%) of the probability mass lies outside of the figure to the right for the within (between)-day plots.

An examination of the time courses of the estimated tuning parameters obtained in this manner indicates that it is often possible to pick out by eye the boundaries between days simply by looking for "jumps" in the tuning parameters. To examine the relative magnitude of within-day and between-day tuning parameter changes, we calculated the standard deviation of the values of each tuning parameter on each day. We also took the mean value of the smoothed parameters for each day and calculated the standard deviation of the day-to-day mean values for each parameter. Fig. 3 C and D show histograms of the two sets of standard deviations obtained in this way for both monkeys. Mean between-day changes in tuning parameters are significantly larger than within-day changes (permutation test, $p < .001$ for both monkeys). This is an insight that will be exploited as we develop the model below.

5. Generative Models for Spike Counts Across Days

We now present two generative, probabilistic models for spike counts observed as a subject performs a discrete BCI task over multiple days. Without loss of generality, we will develop these models in the context of a reaching task, though the algorithm can be directly applied to any discrete decoding scenario. As before, we begin to define the model by defining the prior probability of reaching in one of J directions as

$$y_{d,t} \sim \text{Uniform}\left(\frac{1}{J}\right), \tag{5}$$

which is constant across days and trials. We next define the observation model, $P(\boldsymbol{x}_{d,t}|y_{d,t})$. We model spike counts on any given one of E electrode as conditionally independent of the spike counts on all other electrodes conditioned on reach direction, or $x_{d,e,t} \perp x_{d,i,t}|y_{d,t}, \forall e \neq i$. Using this assumption, we decompose $P(\boldsymbol{x}_{d,t}|y_{d,t} = j)$ as

$$P(\boldsymbol{x}_{d,t}|y_{d,t} = j) = \prod_{e=1}^{E} P(x_{d,e,t}|y_{d,t} = j). \tag{6}$$

The two models we develop correspond to different assumptions on the distributions of spike counts conditioned on reach direction. We refer to one model as the "adaptive Poisson decoder" and define

$$P(x_{d,e,t}|\lambda_{d,e,j}, y_{d,t} = j) = \text{Poisson}(\lambda_{d,e,j}). \tag{7}$$

In Equation (7) we have treated $\lambda_{d,e,j}$ as random variable. As will become apparent immediately below, this is key in the development of the adaptive decoding framework. In addition to the Poisson adaptive decoder, we also introduce a "Gaussian adaptive decoder," where

$$P(x_{d,e,t}|\lambda_{d,e,j}, y_{d,t} = j) = \mathcal{N}(\lambda_{d,e,j}, \sigma^2_{e,j}). \tag{8}$$

In the above equation, we introduce $\sigma^2_{e,j}$ as the conditional variance of spike counts for electrode e and reach direction j, which we model as constant across days. We introduce the Gaussian model for two reasons. First, as will be seen in Sections 6 and 8, it will have computational benefits. Second, the mean and variance of Poisson distributed spikes are constrained to be equal; real neural data often deviates from this and it is possible that a decoder with a Gaussian observation model may outperform one with a Poisson observation model for this reason.

Until this point, the equations above are consistent with standard Poisson and Gaussian naïve Bayes classifiers. We now expand upon these basic models by introducing fixed prior distributions from which tuning parameters for each electrode are drawn anew each day. These priors capture the day-to-day variation in tuning parameters. Specifically, let $\boldsymbol{\lambda}_{d,e} \in \mathbb{R}^{J \times 1}$ be a vector containing the expected number of spike counts on electrode e for day d for each of the J reach directions. For the Poisson adaptive decoder we define

$$\log(\boldsymbol{\lambda}_{d,e}) \sim \mathcal{N}(\boldsymbol{\mu}_e, \Sigma_e), \tag{9}$$

where the log is understood to operate element wise on $\boldsymbol{\lambda}_{d,e}$ and $\boldsymbol{\mu}_e \in \mathbb{R}^{J \times 1}$ and $\Sigma_e \in \mathbb{R}^{J \times J}$ are electrode-specific parameters. For the Gaussian adaptive decoder we define

$$\boldsymbol{\lambda}_{d,e} \sim \mathcal{N}(\boldsymbol{\mu}_e, \Sigma_e). \tag{10}$$

We have overloaded the notation for $\boldsymbol{\mu}_e$ and Σ_e between (9) and (10), but as we proceed in this paper, context will make it clear which parameters we are referring to. Defining the prior over the log of the tuning parameters in (9) ensures a zero probability of the $\boldsymbol{\lambda}_{d,e}$ variables taking on values less than zero, which is required for the Poisson adaptive decoder. Modeling the joint prior distribution of tuning parameters for each electrode as a multivariate Gaussian in both (9) and (10) allows the prior distributions to capture the strong correlation in tuning parameter drift for the same electrode. This correlation can be clearly seen in Fig. 3 panels A and B, where the tuning parameters for the same electrode move up and down across days in a fairly synchronized manner.

6. Decoding across Multiple Days without Retraining

Having defined two models for the generation of spike counts across days in Section 5, we now present recursive Bayesian methods for decoding reach direction from spike counts generated according to each model. In principle, when decoding trial t on day d we could simply integrate out the $\boldsymbol{\lambda}_{d,e}$ variables to form the posterior $P(y_{d,t}|\boldsymbol{x}_{d,t})$. While this would be one method of accounting for the uncertainty in the tuning parameters, it would be suboptimal. Specifically, it ignores the collection of unlabeled spike counts, $\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}$, that the algorithm observes up until trial t on any day.

These unlabeled spike counts are highly informative. Given the observation models (7) and (8), it can be shown that the spike counts on day d, $\boldsymbol{x}_{d,t}$, will cluster according to reach direction, thereby providing information about $\boldsymbol{\lambda}_{d,e}$. In what follows we present principled, recursive methods for jointly estimating the posterior distribution over tuning parameters after each trial, $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$, and the posterior distribution for reach direction, $P(y_{d,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$.

The decoding algorithms for both models proceed in iterations, where one iteration is performed for each trial. For trial t, we begin with the posterior distribution over tuning parameters for the previous trial. For the Gaussian decoder this is $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$, and for the Poisson decoder, we use the posterior over the log tuning parameters, $P(\{\log(\boldsymbol{\lambda}_{d,e})\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$. We then compute $P(y_{d,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$ and $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$ for the Gaussian and $P(\{\log(\boldsymbol{\lambda}_{d,e})\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$ for the Poisson decoder for the present trial in three steps. In the first step for $j = 1, \ldots, J$ for the Gaussian decoder, we compute

$$P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}, y_{d,t} = j) \propto P(\boldsymbol{x}_{d,t}|\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}, y_{d,t} = j)P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}). \quad (11)$$

Above, we have used the independence properties $\boldsymbol{x}_{d,t} \perp \{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}|\boldsymbol{\lambda}_{d,e}$ and $\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E} \perp y_{d,t}$. For the Gaussian adaptive decoder, the right hand side of (11) is a product of multivariate Gaussian distributions and the left hand side can be computed in closed form.

For the Poisson adaptive decoder, we have

$$P(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}, y_{d,t} = j) \propto P(\boldsymbol{x}_{d,t}|\log(\{\boldsymbol{\lambda}_{d,e})\}_{e=1}^{E}, y_{d,t} = j)P(\log(\{\boldsymbol{\lambda}_{d,e})\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}). \quad (12)$$

The right hand side of (12) is a product of Poisson distributions formed from (6) and (7) with a multivariate Gaussian over the logarithm of the tuning parameters obtained from the previous iteration. There is no known analytical method for finding the normalizing constant for this distribution. However, (11) is log concave with respect to $\{\log(\boldsymbol{\lambda}_{d,e})\}_{e=1}^{E}$, and we use Laplace's method to approximate the left hand side of (11) as a multivariate Gaussian. This approximation is a source of significant added computational burden for the Poisson decoder as compared to the Gaussian decoder as it requires a peak finding algorithm to find the mode of $P(\{\log(\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}, y_{d,t} = j)$.

For both algorithms, we next compute

$$P(y_{d,t} = j|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}) \propto P(\boldsymbol{x}_{d,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}, y_{d,t} = j)P(y_{d,t} = j). \quad (13)$$

where $P(\boldsymbol{x}_{d,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}, y_{d,t} = j)$ is found with the normalizing constant for the right hand side of (11) for the Gaussian decoder and with the normalizing constant approximated via Laplace's method for (12) for the Poisson decoder.

Finally, $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$ is computed as

$$P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}) = \sum_{j=1}^{J} P\left(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}, y_{d,t} = j\right) \times P\left(y_{d,t} = j|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}\right), \quad (14)$$

for the Gaussian decoder, and for the Poisson decoder we again work in log space and write this as

$$P(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}) = \sum_{j=1}^{J} P\left(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}, y_{d,t} = j\right) \times P\left(y_{d,t} = j|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t}\right). \tag{15}$$

For both algorithms, (14) and (15) represent a mixture of Gaussians where the two terms on the right hand side of each are obtained from the first two decoding steps just described. In order to avoid an exponential increase in the number of mixture components of the posterior distribution of tuning parameters with increasing iterations of our algorithm, we approximate (11) and (12) as a single Gaussian at each iteration using moment matching, as is done in other hypothesis merging algorithms (Blom and Bar-Shalom (1988)).

7. Improving Robustness Using Outlier Detection

The decoding procedures above are sufficient for decoding data generated according to the two models presented in this work. However, real-world data do not exactly conform to these models, and tuning parameters do change within a day. While small within-day changes to tuning parameters present no problem, large changes, such as may be encountered if the threshold level of an electrode is changed during an experimental session, do.

In particular, if the the number of spike counts, $x_{d,e,t}$, for any given electrode differs substantially from the expected number of spike counts, $\lambda_{d,e,j}$, for one or more reach directions, j, on that electrode, then the term in the observation model for this electrode may dominate the class posterior probability. Persistent inaccuracy in estimating class posterior probabilities will result in inaccurate estimates of tuning parameters, and decode performance can greatly degrade.

It is therefore important to flag any electrodes which are behaving substantially differently than "expected" and increase the uncertainty of the tuning parameters for these electrodes. To formalize this intuition, we can approximate $P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ for each electrode. This is the probability of observing a particular spike count on electrode e given the spike count values for all past trials on all electrodes. For both decoders, this approximation can be obtained in a straight-forward manner as

$$P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}) = \sum_{j=1}^{J} P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}, y_{d,t} = j)P(y_{d,t} = j). \tag{16}$$

Using the same conditional independence properties as in (11), we can write

$$P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}, y_{d,t} = j) = \int P\left(x_{d,e,t}|\boldsymbol{\lambda}_{d,e}, y_{d,t} = j\right) P(\boldsymbol{\lambda}_{d,e}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}) d\boldsymbol{\lambda}_{d,e}, \tag{17}$$

for the Gaussian decoder. $P\left(x_{d,e,t}|\boldsymbol{\lambda}_{d,e}, y_{d,t} = j\right)$ can be obtained from (8) and $P(\boldsymbol{\lambda}_{d,e}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ can be obtained from (14). The integrand in (17) is a product of two Gaussian distributions and the right hand side of (17) can be computed analytically. For the Poisson decoder, we can write

$$P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}, y_{d,t}=j) = \int P\left(x_{d,e,t}|\log(\boldsymbol{\lambda}_{d,e}), y_{d,t}=j\right) P(\log(\boldsymbol{\lambda}_{d,e})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1}) d\boldsymbol{\lambda}_{d,e}. \tag{18}$$

$P\left(x_{d,e,t}|\log(\boldsymbol{\lambda}_{d,e}), y_{d,t}=j\right)$ can be obtained from (7) and $P(\log(\boldsymbol{\lambda}_{d,e})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ can be obtained from (15). The integral in (18) can be approximated using Laplace's method.

After obtaining $P(x_{d,e,t}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$, upper and lower bounds are set on values of $x_{d,e,t}$ for both algorithms such that if the distribution of the number of spike counts for that electrode conforms to the model, observed counts will fall outside of these bounds approximately 1% of the time. When an observed count does fall outside of the bounds, the tuning parameters for that electrode are marginalized out of $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ in (14) for the Gaussian decoder or $P(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ in (15) for the Poisson decoder and effectively "reset" by multiplying the resulting marginalized distribution with a second distribution over just the tuning parameters for the suspect electrode. For the Gaussian decoder this is a Gaussian distribution over $\boldsymbol{\lambda}_{e,j}$, with a mean equal to the marginal mean for electrode e of $P(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E}|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ and covariance Σ_e. For the Poisson decoder this is a Gaussian distribution over $\log(\boldsymbol{\lambda}_{e,j})$ with mean $\boldsymbol{\mu}_e$ and covariance Σ_e; in other words, this is the same prior distribution for the tuning parameters of the electrode the decoder began the day with. Empirically, it was found that selecting the new distributions over suspect electrodes in these two different ways for the Poisson and Gaussian decoders gave the best performance.

After modifying the distributions $P(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ and $P(\log(\{\boldsymbol{\lambda}_{d,e}\}_{e=1}^{E})|\{\boldsymbol{x}_{d,s}\}_{s=1}^{t-1})$ in this way, the decoding algorithms proceed as normal. In this way, normally behaving electrodes are rarely flagged but those which undergo significant within-day tuning parameter changes are flagged and the uncertainty of the tuning parameters for these flagged electrodes is increased so that robust decoding is maintained.

While the spike counts we analyze in this paper are formed by simply counting the number of voltage threshold crossings per electrode, it is common in the BCI literature to add an additional preprocessing step referred to as spike sorting. In this step, the voltage waveform associated with each threshold crossing is analyzed and the spike is assigned to one of a handful of "neural units" for each electrode. This presents a challenge when decoding as neural units may appear or disappear, and the appropriate $\boldsymbol{\lambda}_{e,d}$ variables would need to be added to or removed from the state space. However, the mechanisms described in this Section could potentially be extended if an independent spike sorting algorithm was used to detect the discovery and loss of neural units. In this scenario, a neuron that was lost would simply be marginalized out of (14) or (15) as above, where e would now index neural units instead of electrodes and discovered neurons could be assigned a generic prior distribution for their tuning parameters and added into the decoder in the same way flagged electrodes are reset.

8. Learning the Model Parameters

Both models must be fit to a set of training data collected on a small set of days before they can be used for decoding. To fit the Poisson model we must learn the $\boldsymbol{\mu}_e$ and Σ_e parameters of the underlying prior distributions (9). To fit the Gaussian model, we must learn the $\boldsymbol{\mu}_e$ and Σ_e parameters of (10) as well as the $\sigma_{e,j}$ parameters of (8). Learning the values of these parameters presents us with two challenges. First, the $\boldsymbol{\lambda}_{d,e}$ variables in (7) and (8) are latent and arriving at a formula for the likelihood of the observed data requires integrating over $\boldsymbol{\lambda}_{d,e}$. Given the form of our models, integration over $\boldsymbol{\lambda}_{d,e}$ is not analytically tractable, so we develop expectation-maximization (EM) (Dempster et al. (1977)) algorithms to learn the model parameters.

The second challenge is that $\boldsymbol{\mu}_e$ and Σ_e are parameters for a distribution over *daily* tuning parameters. Therefore, each day of training data, no matter how many individual trials it may contain, essentially gives only one observation with which to learn these parameters. While the value of the $\boldsymbol{\mu}_e$ parameters may be reasonably learned using a small number of training days, learning Σ_e for anything more than a small number of reach directions requires a prohibitive number of training days. This motivates a simplification to our models where we assume the prior distributions for all electrodes share the same covariance matrix, which we will refer to as Σ for both models. This permits us to pool training days across all electrodes when learning Σ.

We now describe the EM algorithms for learning Σ and $\boldsymbol{\mu}_e$ and $\sigma_{e,j}$ for both models. We will refer to the set of days on which training data is collected as $\mathcal{D}_{\text{Train}}$. The observed variables for the EM algorithms are the spike counts for each electrode, $\boldsymbol{x}_{d,t}$, and reach direction, $y_{d,t}$, for each trial for each day $d \in \mathcal{D}_{\text{Train}}$. For day d, we group these observed variables for all trials into the matrices X_d and Y_d, respectively. The latent variables are $\boldsymbol{\lambda}_{d,e}$ for all $d \in \mathcal{D}_{\text{Train}}$, which we group into the variable Λ_d for day d. In the E-step of both algorithms, we wish to compute expectations with respect to:

$$P(\{\Lambda_d\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}}) = \prod_{d\in\mathcal{D}_{\text{Train}}} \prod_{e=1}^{E} P(\boldsymbol{\lambda}_{d,e}|X_d, Y_d) \tag{19}$$

using Σ, $\boldsymbol{\mu}_e$ and $\sigma_{e,j}$ from the previous M-step. For the Gaussian adaptive decoder, this distribution can be computed exactly as the right hand side of 19 is simply the product of multivariate Gaussians. However, for the Poisson decoding model, the Gaussian prior (9) and product of Poisson observation models (7) renders computing the posterior distribution (19) analytically intractable. However, $P(\log(\boldsymbol{\lambda}_{e,d})|X_d, Y_d)$ is log-concave with respect to $\log(\boldsymbol{\lambda}_{e,d})$ and we use Laplace's method to approximate $P(\{\log(\Lambda_d)\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$ as a multivariate Gaussian. Note this approximation renders our EM algorithm only an *approximate* EM algorithm, and adds additional computational cost as a peak finding method must again be used to find the mode of $P(\{\log(\Lambda_d)\}_{d\in\mathcal{D}_{\text{Train}}}|\{X_d, Y_d\}_{d\in\mathcal{D}_{\text{Train}}})$.

In the M-step of the EM algorithms we can compute the parameter update for $\hat{\boldsymbol{\mu}}_e$ as:

$$\hat{\boldsymbol{\mu}}_e = \frac{1}{|\mathcal{D}_{\text{Train}}|} \sum_{d\in\mathcal{D}_{\text{Train}}} \mathbb{E}_{P'}\left[\boldsymbol{\lambda}_{d,e}\right] \tag{20}$$

for the Gaussian decoding model and

$$\hat{\boldsymbol{\mu}}_e = \frac{1}{|\mathcal{D}_{\text{Train}}|} \sum_{d \in \mathcal{D}_{\text{Train}}} \mathbb{E}_{P'} \left[\log(\boldsymbol{\lambda}_{d,e}) \right] \tag{21}$$

for the Poisson decoding model, where P' indicates $P(\boldsymbol{\lambda}_{d,e}|X_d, Y_d)$ for the Gaussian decoding model and the Gaussian approximation to $P(\log(\boldsymbol{\lambda}_{d,e})|X_d, Y_d)$ for the Poisson decoding model obtained in the E-step. The notation $| \cdot |$ indicates the number of elements in a set. The updates for $\hat{\Sigma}$ are

$$\hat{\Sigma} = \frac{1}{E \cdot |\mathcal{D}_{\text{Train}}|} \sum_{d \in \mathcal{D}_{\text{Train}}} \sum_{e=1}^{E} \mathbb{E}_{P'} \left[(\boldsymbol{\lambda}_{d,e} - \hat{\boldsymbol{\mu}}_e)(\boldsymbol{\lambda}_{d,e} - \hat{\boldsymbol{\mu}}_e)^T \right] \tag{22}$$

for the Gaussian observation model and

$$\hat{\Sigma} = \frac{1}{E \cdot |\mathcal{D}_{\text{Train}}|} \sum_{d \in \mathcal{D}_{\text{Train}}} \sum_{e=1}^{E} \mathbb{E}_{P'} \left[(\log(\boldsymbol{\lambda}_{d,e}) - \hat{\boldsymbol{\mu}}_e)(\log(\boldsymbol{\lambda}_{d,e}) - \hat{\boldsymbol{\mu}}_e)^T \right] \tag{23}$$

for the Poisson observation model.

Finally, the update for $\hat{\sigma}_{e,j}$ for the Gaussian decoding model is

$$\hat{\sigma}_{e,j} = \frac{\sum_{d \in D_{\text{Train}}} \sum_{t: y_{d,t} = j} \mathbb{E}_{P'} \left[(x_{d,e,t} - \lambda_{d,e,j})^2 \right]}{\sum_{d \in D_{\text{Train}}} \sum_{t: Y_d(t) = j} (1)}. \tag{24}$$

For both models the expectants in equations (20), (21), (22), (23) and (24) can easily be obtained from the first and second moments of P'.

When determining when to stop the the EM algorithm, the log-likelihood of the spike counts conditioned on reach direction, $\log \left(P(\{X_d\}_{d \in \mathcal{D}_{\text{Train}}} | \{Y_d\}_{d \in \mathcal{D}_{\text{Train}}}) \right)$, is monitored. For the Gaussian decoding model, this can be calculated exactly. For the Poisson decoding model this can be approximated as sum of the normalizing constants found with Laplace's method for the $P(\log(\boldsymbol{\lambda}_{d,e})|X_d, Y_d)$ terms approximated in the E-step. The EM algorithms for the Poisson and Gaussian models were developed at different points in time and the stopping rules for each differ slightly. Specifically, for the Poisson model, the number of EM iterations is determined by monitoring the approximation of the log-likelihood of the observed counts conditioned on reach direction, $\log P(\{X_d\}_{d \in \mathcal{D}_{\text{Train}}} | \ \{Y_d\}_{d \in \mathcal{D}_{\text{Train}}})$, returned after each E-step and performing one more M-step after this log-likelihood fails to increase. For the Gaussian model, the number of EM iterations is stopped after the log-likelihood of the observed counts conditioned on reach direction increases by less than a value of 1E-10 between successive iterations.

In practice, the EM algorithms presented here must be initialized with specific initial parameter values. For the Poisson model, the EM algorithm is initialized using the mean log counts conditioned on reach direction for the $\boldsymbol{\mu}_e$ parameters and Σ is initialized as the identity matrix. For the Gaussian model, the EM algorithm is initialized using the mean counts conditioned on reach direction over all days for the $\boldsymbol{\mu}_e$ parameters and Σ is initialized as the empirical covariance matrix for the daily mean counts conditioned on reach direction[1]. The $\sigma_{j,e}$ parameters are initialized as the conditional variances in spike counts for electrode e and reach direction j across all days, where the shift in mean counts on each day is considered when calculating these values.

9. Results

We now apply the adaptive Gaussian and Poisson decoders to the same datasets introduced in Section 2 and analyzed in Sections 3 and 4. For both datasets, we trained the decoders on the first 10 days of data and tested on the remaining 31 (26) days in each.

As in Section 3, the goal of the decoders was to predict reach direction. As the data was prerecorded, trials on all days were labelled with the reach direction of the monkey. However, labels on test days were not made available to the adaptive decoders and were only referenced later to compare predicted to true reach directions.

Before training the algorithms, we first removed any electrodes with fewer than 2 threshold crossings on average in the spike count window, as was done for the static decoder in Section 3. As this was done using only training data, this is a step that can be carried out in a clinical setting.

We then learned the $\boldsymbol{\mu}_e$ and Σ parameters for the Poisson decoder and the $\boldsymbol{\mu}_e$, Σ and $\sigma_{e,j}$ parameters for the Gaussian decoders with the EM algorithms described in Section 8. For Monkey M, the fitting procedure for both models reached convergence; for Monkey I the fitting procedure for the Poisson model reached convergence, while that for the Gaussian decoder did not reach convergence before the limit of 1000 EM iterations was reached.

After training the models, we decoded reach direction on the remaining 31 (26) days of data using the decoding algorithms described in Section 6. We compared the performance of these adaptive decoders with the retrained and static Poisson naïve Bayes decoders introduced in Section 3. As the reader will remember, the retrained decoder was trained on trials 1-400 on each test day and then tested on the remaining trials of each day. When testing the performance of the remaining three decoders, we also omitted trials 1-400 on each day, starting them at trial 401. This is important as it ensures that any trials the adaptive decoders must use to "burn-in" and learn the correct tuning parameters for a day are included when comparing performance against the retrained method.

Fig. 4 shows the mean and 95% score confidence intervals (Agresti and Coull (1998)) of the daily accuracies of all four decoders for both datasets. When calculating overall accuracy of each decoder we want a statistic that reflects the stability of the performance of each algorithm across all days. To achieve this, we first calculate the mean decode accuracy of a decoder on each test day and then measure overall accuracy as the mean of these mean daily performance values. This gives each day equal weighting in this statistic, no matter how many individual trials are performed on particular days. The 95% confidence interval for the overall accuracy calculated in this way of the static decoder was $63 \pm 4\%$ $(61 \pm 5\%)$. The overall accuracy of the retrained decoder was $73 \pm 2\%$ $(77 \pm 3\%)$. The overall accuracy of the Poisson adaptive decoder was $74 \pm 2\%$ $(70 \pm 3\%)$ and that of the Gaussian

1. The Gaussian decoder was developed after the Poisson decoder. Future work may initialize Σ for the Poisson decoder in a manner similar to that of the Gaussian.

adaptive decoder was $75 \pm 1\%$ ($69 \pm 3\%$), demonstrating that both adaptive decoders give a significant improvement over the static decoder. Most importantly, an examination of Fig. 4 reveals that both adaptive decoders consistently produced stable decode performance without retraining. The only day this is not the case for is day 23 for monkey I, for which even the retrained decoder shows poor performance. As explained in Section 3, we believe there may have been a significant resetting of threshold values across many of the electrodes in the middle of the experimental session during this day.

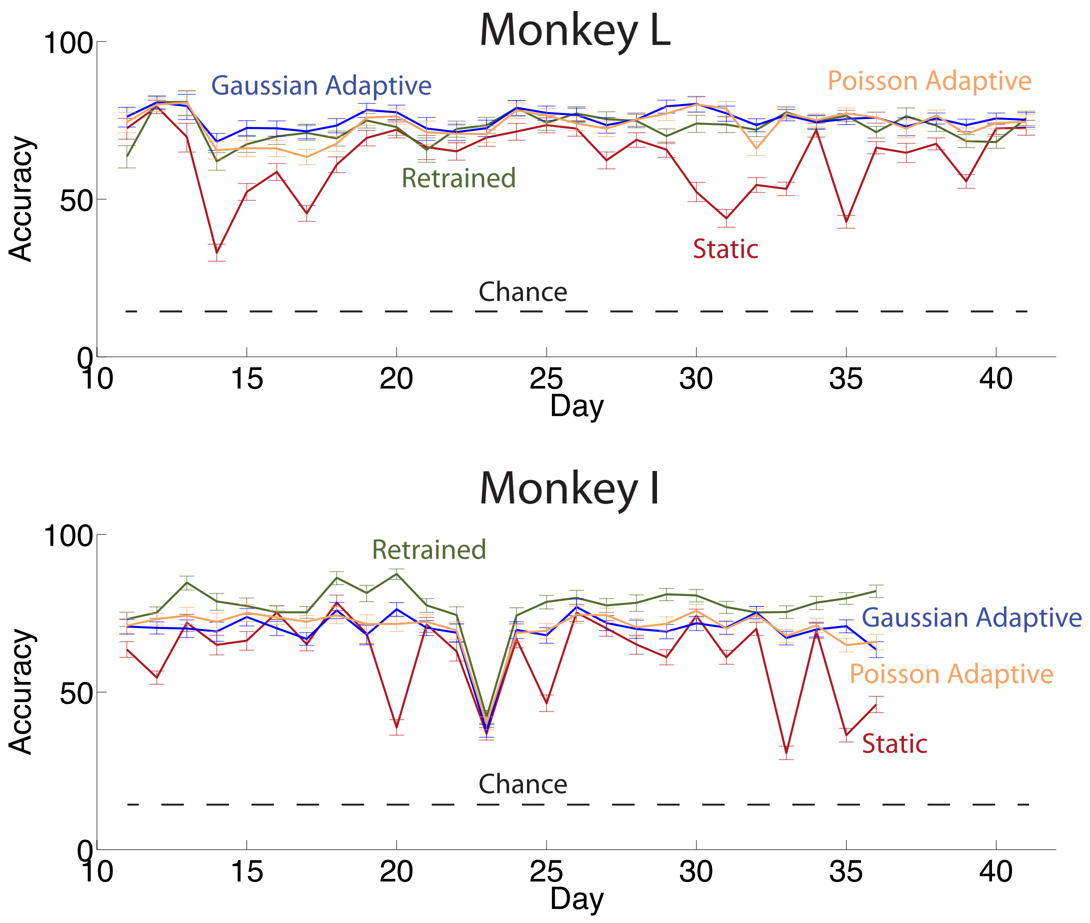
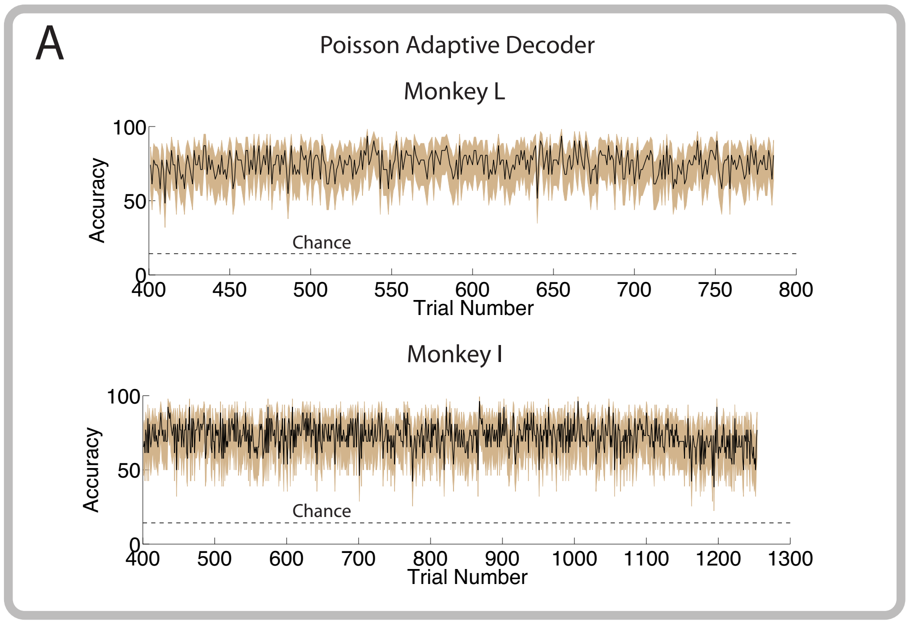


Figure 4: A: Mean daily performance and 95% score confidence intervals of the static, retrained, adaptive Poisson and Adaptive Gaussian decoders for both monkeys over all test days.

Unlike the static or retrained decoders, the adaptive decoders perform inference on neural tuning parameters, $\lambda_{d,e,j}$, through a decoding day, which means we might reasonably expect to see the performance of these decoders change across time. For example, we might expect to find evidence of a "burn-in" period at the beginning of the day where accuracy is lower as the adaptive algorithms learn the correct parameters for a day. There were 385

(854) or more trials during each test day, and using all 31 (26) test days we calculated the probability that trial $t \in [1, \ldots, 385]$ ($t \in [1, \ldots, 854]$) was decoded correctly on any given day. The mean and 95% score confidence intervals of the decode accuracy with increasing trial number is shown in Fig. 5 A and B for both decoders and monkeys. While the results here are not conclusive, the traces in Fig. 5 may provide evidence of a short (< 50 trials) burn-in period, with all decoders starting a decode day well above chance performance.



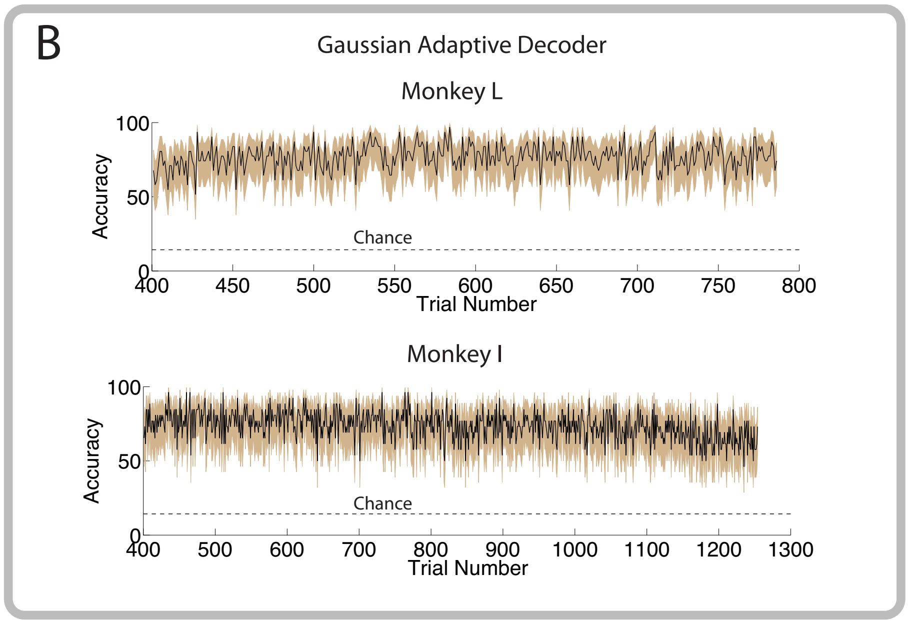


Figure 5: A: Probability that trial t was calculated correctly on any given test day for the Poisson (A) and Gaussian (B) decoders for both monkeys. Shaded region shows 95% score confidence intervals. On each day, the decoders were started on trial 401.

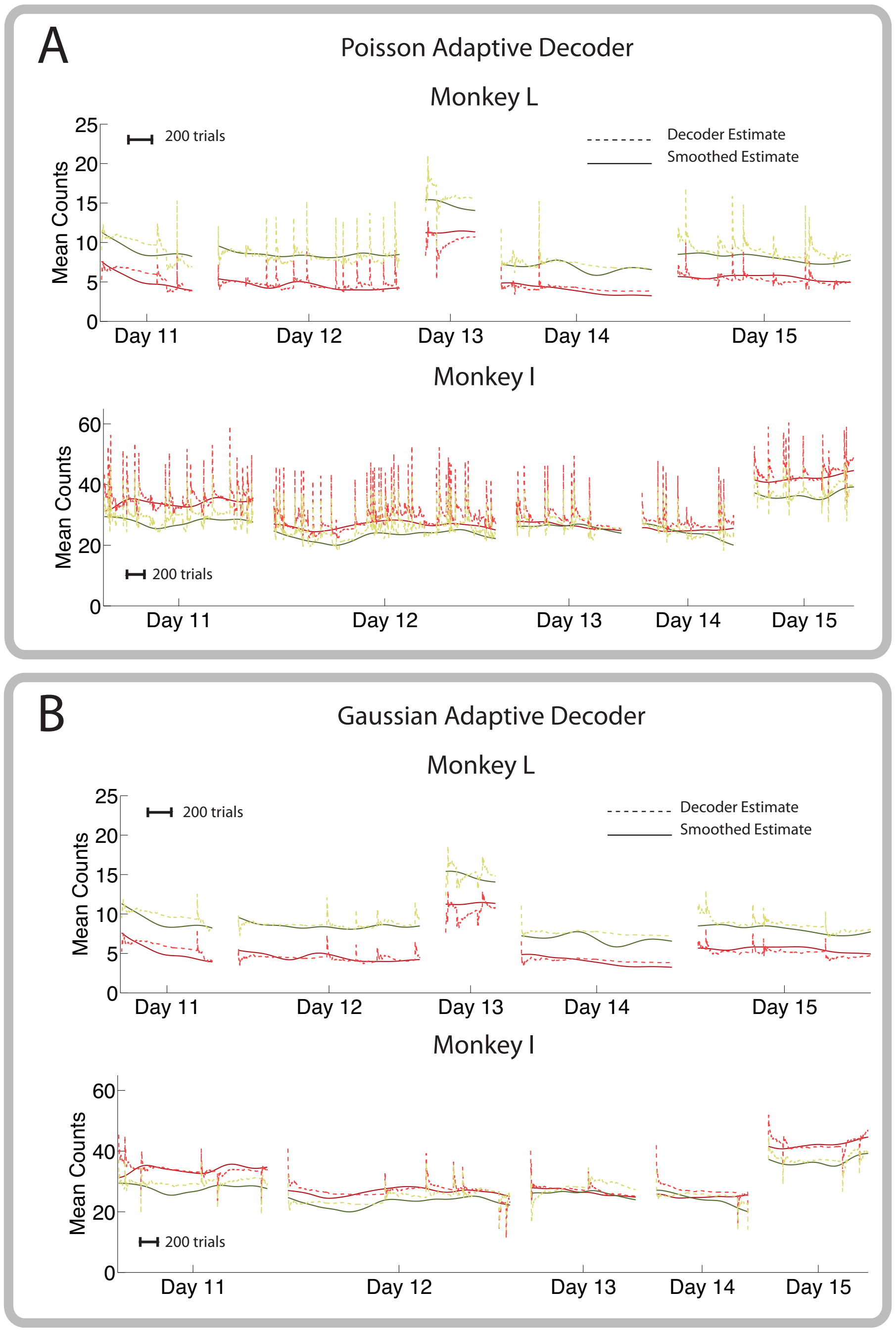


Figure 6: Estimates of two tuning parameters for the same electrode obtained with supervised (Gaussian) smoothing and from estimates produced by the adaptive Poisson (A) and Gaussian (B) decoders over the first 5 days of test data from both monkeys. When comparing between $\hat{\lambda}_{d,e,j}$ estimates for the same monkey, the electrode and tuning parameters selected are the same.

As the adaptive decoders perform inference on tuning parameters, $\lambda_{d,e,j}$, throughout an experimental day, we can examine the estimates they return. Strictly speaking, the decoders do not return point estimates for $\hat{\lambda}_{d,e,j}$ but posterior distributions over $\{\lambda_{d,e,j}\}_{e=1}^{E}$ for each trial, $P(\{\lambda_{d,e,j}\}_{e=1}^{E} | \{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$, and we chose to use the posterior means $\mathbb{E}[\lambda_{d,e,j} | \{\boldsymbol{x}_{d,s}\}_{s=1}^{t}]$ calculated from an approximation to $P(\lambda_{d,e,j} | \{\boldsymbol{x}_{d,s}\}_{s=1}^{t})$ returned from both algorithms as our point estimates for $\hat{\lambda}_{d,e,j}$. Fig. 6 displays traces of estimates for two tuning parameters for one electrode over the first five days of test data returned from both decoders for both monkeys. When comparing between $\hat{\lambda}_{d,e,j}$ estimates for the same monkey, the electrode and tuning parameters selected are the same. We compare the estimates produced by the adaptive decoders to those obtained with the same Gaussian smoothing (kernel width of 100 trials) used in Section 4. However, unlike the results shown in Section 4 where days were concatenated together before performing smoothing, each day was processed separately here.

An examination of Fig. 6 reveals some interesting things. First, note that the true tuning parameters (as measured with the supervised smoothing method) can be substantially shifted from one day to the next and the estimates of $\hat{\lambda}_{d,e,j}$ for both decoders are able to track these large between-day changes.

The behavior of the decoders within a day is also interesting. The sudden jumps in parameter estimates produced by the adaptive decoder occur when a spike count for this electrode caused the electrode to be flagged as behaving erratically and the distribution for the tuning parameters for this electrode was reset. Bounds on spike counts were set so that electrodes behaving according to the model were reset 1% of the time, or about 1 in every 100 trials, and the frequency of resets observed in some panels of Fig. 6 is therefore not surprising, though the frequency seen in the lower part of panel A is somewhat high. We have observed this behavior on other electrodes (data not shown), and it is likely attributable to the distribution of spike counts on these electrodes not conforming to our modeling assumptions. We know that while this makes the visible traces of the $\hat{\lambda}_{d,e,j}$ values less visually appealing, the purpose of these algorithms is ultimately not to estimate the set of $\hat{\lambda}_{d,e,j}$ values with time but to produce stable decoder performance across time without supervised retraining, which both algorithms do well.

10. Conclusion

We have proposed two semi-supervised classifiers for discrete intracortical brain computer interface (BCI) which maintain stable decoder performance on two prerecorded datasets over a period of 31 (26) days. To our knowledge, this is the first demonstration of stable intracortical BCI classification across many days. While this is the main contribution of this paper, we conclude by considering the implications of the results presented in this work and outlining directions of future work.

In Section 3, we demonstrated the performance of a standard BCI decoder which was trained on an initial set of 10 days of data and then used to decode 31 (26) more days of data with the parameters learned in training fixed. We found that while performance from day-to-day could be highly variable, mean daily decode accuracy did not tend to decline with time. Specifically, the 95% confidence interval for the slope of a linear fit of mean daily performance for monkey M was (-0.005 %/day, 0.005 %/day) and (-0.012573 %/day, 0.001665 %/day) for monkey I. This is a very interesting finding, and it is tempting to speculate that some constrained process, such as micromovements of the recording array relative to the brain tissue, is responsible for signals drifting in this manner. This suggests a future direction of research for basic scientists and those designing physical BCI sensors.

In Section 4 we characterized the relative amount of drift in tuning parameters between and within days and found mean between-day changes in tuning parameters were significantly larger than within-day changes (permutation test, $p < .001$ for both monkeys). This motivated a simplification in the specification of our generative models described in Section 5 where we modeled tuning parameters as variable between days but fixed within a day. The performance results of the adaptive decode algorithms given in Section 9 indicate that this assumption, though a simplification of reality, is adequate to permit stable decoding. It is possible to derive decoding and parameter learning algorithms for models which assume parameters randomly walk between trials. However, this assumption makes inference in the E-step of the EM algorithms significantly more challenging as the latent $\boldsymbol{\lambda}_{d,e}$ variables must now be indexed by trial (as they are assumed to change from trial to trial) and form a time series, which greatly increases the dimensionality of the space that inference must be performed over. Nonetheless, it is possible and we may present the results of these more complicated models in future work.

Both the Poisson and Gaussian adaptive decoders produced stable classification over periods of many days. As mentioned in Sections 6 and 8, there is a significant computational burden associated with performing the Laplace approximations necessitated by the use of Poisson observation models for the Poisson adaptive decoder. A deployed BCI system must run in real-time, meaning it must quickly classify brain signals as they are recorded, and a decoder with less computational cost is to be preferred. For this reason, we would suggest that real world BCI systems use the Gaussian adaptive decoder.

Finally, we note that the next step in validating these algorithms requires online, closed loop decoding. This is an important step in validating the performance of any BCI algorithm and especially one that adapts to changing brain signals. In real, closed loop use of a BCI, a subject may actively engage cognitive strategies to attempt to adapt his or her brain signals to improve BCI performance. The interplay of the subject's cognitive adaption with the adaption performed by the algorithms proposed here cannot be assessed with prerecorded data and could be significant for determining the real world applicability of this work. Therefore, we look forward to finding collaborators excited to apply these decoders in a closed loop setting in the near future.

Acknowledgments

The author would like to gratefully acknowledge Cynthia A. Chestek, Vikash Gilja, Paul Nuyujukian, Stephen I. Ryu, and Krishna V. Shenoy for making these datasets available for his use.

We would also like to acknowledge support for this project from NIH-NICHD-CRCNS, NIH-NINDS-CRCNS, NIH Pioneer, NIH Grant T90 #DA022762, NIH Grant R90 DA023426-06, NIH-NICHD-CRCNS-R01, DARPA, BWF, CRPF, McKnight, Sloan, NSFGR and NDSEG Fellowships, MSTP, Soros and Stanford.

References

A. Agresti and B.A. Coull. Approximate is better than "exact" for interval estimation of binomial proportions. *American Statistician*, pages 119–126, 1998.

Justin Baker, William Bishop, Spencer Kellis, Todd Levy, Paul House, and Bradley Greger. Multi-scale recordings for neuroprosthetic control of finger movements. In *Proc. Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBC 2009*, pages 4573–4577, 2009. doi: 10.1109/IEMBS.2009.5332692.

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.

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, 2007.

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. *Conf Proc IEEE Eng Med Biol Soc*, 2009:3369–3375, 2009. doi: 10.1109/IEMBS.2009.5332822. URL `http://dx.doi.org/10.1109/IEMBS.2009.5332822`.

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.

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 Comput*, 16(5):971–998, May 2004. doi: 10.1162/089976604773135069. URL `http://dx.doi.org/10.1162/089976604773135069`.

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.

Karunesh Ganguly and Jose M Carmena. Emergence of a stable cortical map for neuroprosthetic control. *PLoS Biol*, 7(7):e1000153, Jul 2009. doi: 10.1371/journal.pbio.1000153. URL `http://dx.doi.org/10.1371/journal.pbio.1000153`.

V. Gilja, C. Chestek, I. Diester, J. Henderson, K. Deisseroth, and K. Shenoy. Challenges and opportunities for next-generation intra-cortically based neural prostheses. *Biomedical Engineering, IEEE Transactions on*, (99):1–1, 2011.

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, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

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*, pages 1–43, 2011.

S. Musallam, BD Corneil, B. Greger, H. Scherberger, and RA Andersen. Cognitive control signals for neural prosthetics. *Science*, 305(5681):258–262, 2004.

P. Nuyujukian, J. Kao, J.M. Fan, S. Stavisky, S.I. Ryu, and K.V. Shenoy. A high-performance, robust brain-machine interface without retraining. In *Frontiers in Neuroscience. Conference Abstract: Computational and Systems Neuroscience (COSYNE)*, number III-65, Salt Lake City, UT, 2012.

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.

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. *Biomedical Engineering, IEEE Transactions on*, 54(11):2037–2050, 2007.

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. doi: 10.1038/nature04968. URL `http://dx.doi.org/10.1038/nature04968`.

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.

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. doi: 10.1038/nature06996. URL `http://dx.doi.org/10.1038/nature06996`.