DECODING NEURAL ACTIVITY USING KALMAN FILTER

A PREPRINT

Agam Tomar
Department of Electrical and Computer Engineering
University of California Los Angeles
agamtomar@engineering.ucla.edu

July 19, 2019

ABSTRACT

Brain-machine interface (BMI) systems converts neural signals from motor regions of the brain into control signals to guide prosthetic devices. The paper reviews a BMI system, its categorization and decoder algorithms which are critical to BMI system's performance. Kalman filter is investigated in conversion of neural signals into prosthetic control signals by implementing it on a real data set and comparing performance with previous decoding algorithms.

***K*eywords** brain-machine interface (BMI) · decode algorithm · neural signal processing

1 Introduction

Neural decoding is used to make predictions about variables in the outside world from the activity recorded in brain. Earlier research includes prediction of movements based on activity in motor cortex [1], predict decisions based on activity in prefrontal and parietal cortices, and predict locations based on activity in the hippocampus [2]. When predicting a continuous variable, decoding is simply a regression problem and when predicting a discrete variable, decoding is simply a classification problem. Thus, there are many methods that can be used for neural decoding. However, despite the recent advances in machine learning techniques, it is still common to use traditional methods such as linear regression. Using modern machine learning tools for neural decoding would likely significantly boost performance, and might allow deeper insights into neural function.

This paper investigates the use of Kalman filter as a decoding algorithm and compares its performance with earlier decoders (Wiener filter).

2 Related Work

2.1 Brain-Machine Interface

Millions of people worldwide suffer from motor-related neurological injury or disease, which in some extreme cases, results in loss of communication ability [3, 4]. For people with lost motor function, brain-machine interfaces (BMIs), also known as neural prostheses or brain-computer interfaces(BCIs), have the potential to increase quality of life and enable greater interaction with the world. Over the past two decades, significant progress has been made towards realizing clinically viable BMI systems.

As illustrated in Figure 1, BMI systems comprise three major components: 1) sensors recording neural activity, typically from motor cortical regions of the brain; 2) a decoder, which translates the neural recordings into control signals; and 3) a prosthesis, such as a computer cursor on a screen or a robotic arm, controlled by the decoder.

Many challenges still remain in achieving clinically viable BMI systems, including: 1) increasing BMI performance and robustness; 2) increasing the functional lifetime of implanted sensors; 3) replacing wires with wireless data telemetry and wireless powering; and 4) improving BMI ease of use, so that constant technician supervision is not required [5].

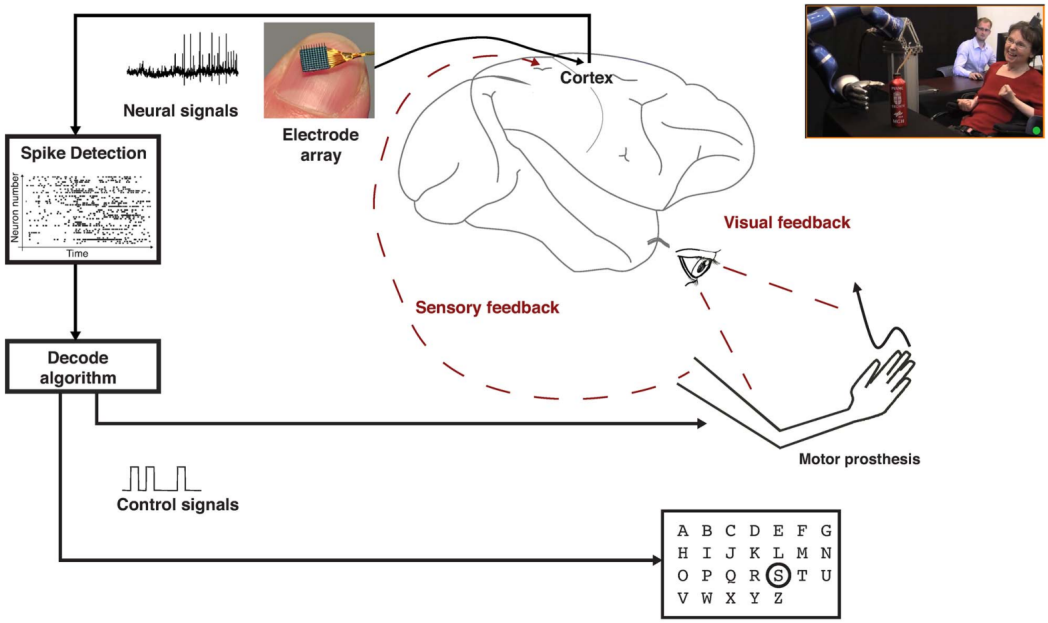

Figure 1: This figure is from [5] providing overview of BMI systems. In an intracortical BMI system, neural signals are recorded from electrode arrays typically implanted in motor cortical regions of the brain. The raw neural signals are then passed through a spike detection algorithm, such as threshold detection,where a spike is detected if the measured electrode voltage crosses a preset threshold value. In the"spike detection"block, a black dot denotes that an action potential was measured. The neural spiking data are then sent to a decode algorithm, which outputs control signals (e.g., a digital signal) that guide a prosthetic device. The movement of the prosthetic device is observed by the subject, which closes a feedback loop.

As decoder is integral to BMI performance and clinical viability, this project focuses on improving decode algorithm design to provide subjects with high-fidelity neural control of a prosthetic device.

2.2 Decoders

The decode algorithms, which translates recorded neural population activity into prosthesis control signals, is essential for high-performance BMI systems. Historically, decoder designs were inspired by neuroscientific views of motor cortex as well as by linear estimation, statistical inference, and neural network theory. BMI decode algorithms are trained with simultaneous observations of real arm or prosthesis kinematics and neural population activity. Figure 3 provides a categorization of BMI algorithms adopted from [5].

1. Population vector (PV): This algorithm was proposed by *Georgopoulos et al.* which is based on a neurophysical result: under certain conditions, the cosine of the reach direction can, in part, describe the firing rate of neurons in macaque motor cortex. The PV algorithm has been used in several BMI systems [6, 7, 8].
2. Wiener Filter: The Wiener filter was a seminal contribution in estimation theory, helping to bring a statistical point of view into communication and control theory [9]. Both Wiener [10] and Kolmogorov [11] independently developed filtering theory in which a noisy sequence of observations $\mathbf{y}_1, ..., \mathbf{y}_k$ is used to calculate a linear estimate of a signal $\mathbf{x}_k$, given by $\tilde{\mathbf{x}}_k = \sum_{j=1}^{k} \mathbf{L}_k^T \mathbf{y}_j$, where $\mathbf{L}_j \in \mathbb{R}^{NxM}$. In Wiener-Kolmogorov filtering theory, the goal is to learn parameters $\mathbf{L}_1, ..., \mathbf{L}_k$ such that the squared error in predicting $\mathbf{x}_k$ is minimized. The major difference between PV algorithms and Wiener-Kolmogorov approach is the incorporation of neural history $\mathbf{y}_{k-1}, \mathbf{y}_{k-1,...}$ into the regression problem. This filter was used in first BMI clinical studies with intracortical electrode arrays, demonstrating that a human could control a computer cursor and perform rudimentary actions with a multi-jointed robotic arm [12].
3. Kalman Filter: Kalman filter is a recursive algorithm that estimates the current state of a dynamical system given an observation of the output of the dynamical system and the previous state estimate [13]. In general, the state-of-the-art BMI systems using Kalman filtering model the prosthesis kinematics as the sate of a

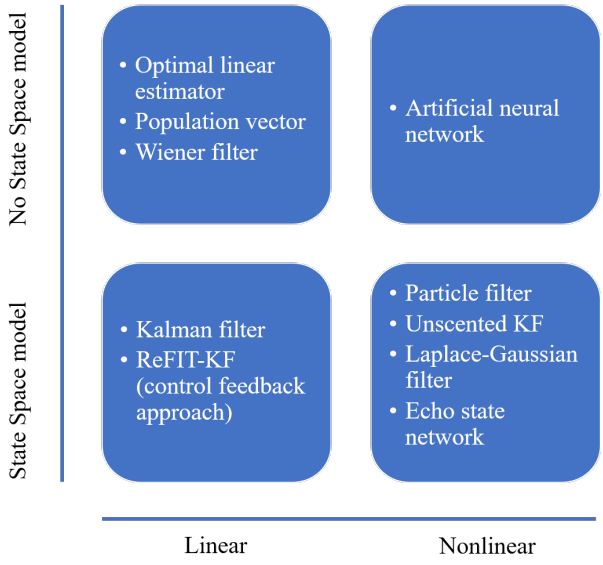

Figure 2: This figure is adopted from [5] showing the categorization of BMI Algorithms

linear dynamical system with certain dynamical update laws. Next section discusses the modeling and implementation of Kalman filter to decode neural activity in detail.

2.3 Kalman Filter

In 1960, Rudolf Emil Kalman published a seminal paper describing a solution to discrete-data filtering problem. The Kalman filter is a recursive algorithm that estimates the current state of a dynamical system given an observation of the output of the dynamical system and the previous state estimate [13]. This recursive algorithm served as a cornerstone in finite-time and non-stationary analyses, particularly in the area of autonomous and assisted navigation [14, 15]. This project discussed the modeling of prosthesis kinematics using Kalman filter as the state of a linear dynamical system with certain dynamic update laws. In the dynamical model, it is assumed that the kinematics obey physical laws and smooth over time [1].

In 2003, *Wu et al.* proposed a Kalman filter technique to estimate the kinematics of a prosthetic device given observations of the neural population activity $\mathbf{y}_k$ [16]. The dynamical model proposes that the kinematics of the prosthesis $\mathbf{x}_k$ are the state of a linear time-invariant dynamical system, while the neural activity $\mathbf{y}_k$ is the output of the dynamical system. The state and the output process are both modeled to have Gaussian noise. Therefore, the system can be written as

$$\begin{aligned} \mathbf{x}_k &= \mathbf{A}\mathbf{x}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \mathbf{q}_k \end{aligned} \tag{1}$$

with $\mathbf{w}_k \sim \mathbb{N}(0, \mathbf{W})$ and $\mathbf{q}_k \sim \mathbb{N}(0, \mathbf{Q})$. Since the sequences $\mathbf{x}_{k=1,\ldots,K}$ and $\mathbf{y}_{k=1,\ldots,K}$ are observed in the training set while $\mathbf{w}_k$ and $\mathbf{q}_k$ are zero mean terms, $\mathbf{A}$ and $\mathbf{C}$ can be learned via least squares regression: $\mathbf{A} = \mathbf{X}_{[2:K]}\mathbf{X}_{[1:K-1]}^T(\mathbf{X}_{[1:K-1]}\mathbf{X}_{[1:K-1]}^T)^{-1}$ and $\mathbf{C} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$. After learning $\mathbf{A}$ and $\mathbf{C}$, $\mathbf{W}$ is calculated as the sample covariance of the residuals $\mathbf{X}_{[2:K]} - \mathbf{A}\mathbf{X}_{[1:K-1]}$ while $\mathbf{Q}$ is analogously the sample covariance of the residuals $\mathbf{Y} - \mathbf{C}\mathbf{X}$. Given $\mathbf{A}, \mathbf{W}, \mathbf{C}, \mathbf{Q}$ as well as an initial state condition, $\mathbf{x}_0$ (which is often set to be zero), the Kalman filter recursively estimates the current state $\tilde{\mathbf{x}}_k$, given the current neural observation $\mathbf{y}_k$, and the previous state estimate $\tilde{\mathbf{x}}_{k-1}$ [13].

A benefit in modeling a dynamical update law for the kinematic variables is the ability to enforce that the prosthesis movements obey physical kinematic laws. For example, if $\mathbf{x}_k = [\mathbf{p}_k^T \mathbf{v}_k^T]^T$, then the $\mathbf{A}$ matrix can be additionally designed such that the position obeys $\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t$. Further, the $\mathbf{A}$ matrix provides a measure of smoothing or low-pass filtering over the kinematic variables. This is important for ensuring that the kinematics are not discontinuous or jarring to the subject controlling the prosthesis. The Kalman filter also casts BMI systems into a Bayesian framework, where it is now possible to model noise processes, effectively weighting neurons based on modeled noise properties.

However, one potential limitation of Kalman filtering is that the state and observation noise processes are typically modeled to be Gaussian, which is an oversimplified assumption.

3 Methodology

3.1 Problem Formulation

Neural prosthesis translates neural activity from the brain into control signals for guiding prosthetic devices using BMI algorithms. Decoding models are a critical part of BMI algorithms which usually account for its performance and clinical translation. All of the decoding models prior to Kalman filter are *static* and do not incorporate *dynamics*. As neural data varies with time, we need models which can capture temporal relationship between data points (i.e., a *dynamical* model, as given by some state-space model).

3.1.1 Linear Dynamical System (LDS)

A *dynamical system* is a system in which a function describes the time dependence of a point in a geometrical space. A *Linear dynamical systems* is a dynamical systems whose evaluation functions are linear.

At time step $k = 1, ..., K$, let:

- $\mathbf{x}_k \in \mathbb{R}^M$ be the state variable (e.g., arm position or velocity)
- $\mathbf{y}_k \in \mathbb{R}^N$ be the observation (e.g., spike count vector)

Then, the linear dynamical system is composed of two models.

The state process:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{w}_k \\ \mathbf{x}_1 &\sim \mathbb{N}(\nu, \mathbf{V}) \end{aligned} \tag{2}$$

The observation process:

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{q}_k \tag{3}$$

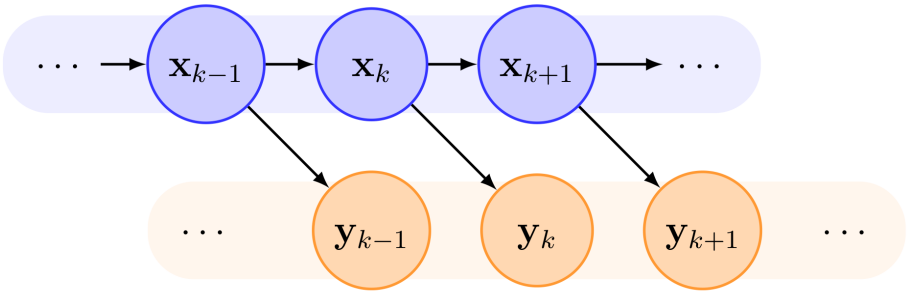

Figure 3: Graphical model of Linear Dynamical System

Here, $\mathbf{w}_k \sim \mathbb{N}(0, \mathbf{W})$ and $\mathbf{q}_k \sim \mathbb{N}(0, \mathbf{W})$. Further, we assume that $\mathbf{w}_k$ and $\mathbf{q}_k$ are independent noise. That is they are independent of each other ($\mathbf{w}_k \perp \mathbf{q}_k$), they are independent over time ($\mathbf{w}_k \perp \mathbf{w}_{k-1}$) and they are independent of the state up to and including the current time and observations in the past ($\mathbf{w}_k, \mathbf{q}_k \perp \mathbf{x}_k, \mathbf{x}_{k-1}, ...$ and $\mathbf{w}_k, \mathbf{q}_k \perp \mathbf{y}_k, \mathbf{y}_{k-1}, ...$). Figure 3 shows the graphical model of linear dynamical system. With the assumptions of $\mathbf{w}_k$ and $\mathbf{q}_k$, the state and observation processes can be rewritten as:

The state process:

$$\begin{aligned} \mathbf{x}_k|\mathbf{x}_{k-1} &\sim \mathbb{N}(\mathbf{A}\mathbf{x}_{k-1}, \mathbf{W}) \\ \mathbf{x}_1 &\sim \mathbb{N}(\nu, \mathbf{V}) \end{aligned} \tag{4}$$

The observation process:

$$\mathbf{y}_k|\mathbf{x}_k \sim \mathbb{N}(\mathbf{C}\mathbf{x}_k, \mathbf{Q}) \tag{5}$$

The model parameters of this linear dynamical system can be summarized as $\theta = \mathbf{A}, \mathbf{W}, \nu, \mathbf{V}, \mathbf{C}, \mathbf{Q}$.

3.2 Training parameters of a linear dynamical system

Our goal is to learn θ from the training data and then use the model for neural decoding.

For BMI applications, we consider that the values of the state variables $\mathbf{x}_k$ are known during training and θ is estimated using the expectation maximization (EM) algorithms which solves the following maximization (subject to local optima):

$$Maximize \quad P(\{\mathbf{x}\}, \{\mathbf{y}\}|\theta) \qquad w.r.t \quad \theta \tag{6}$$

For decoding arm trajectories from neural activity, the $\mathbf{x}_k$ represents the state of the arm, and are typically known during training. The corresponding $\mathbf{y}_k$ are the observed neural activity.

Given the training data sequences, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_K$ and corresponding neural activity $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_K$, we can fit the maximum-likelihood parameters of a linear dynamical system model.

$$\begin{aligned}
\mathbf{A} &= \left(\sum_{k=2}^{K} \mathbf{x}_k \mathbf{x}_{k-1}^T\right)\left(\sum_{k=2}^{K} \mathbf{x}_k \mathbf{x}_{k-1}^T\right)^{-1} \\
\mathbf{W} &= \frac{1}{K-1}\sum_{k=2}^{K}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1})(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1})^T \\
\mathbf{C} &= \left(\sum_{k=1}^{K} \mathbf{y}_k \mathbf{x}_k^T\right)\left(\sum_{k=1}^{K} \mathbf{x}_k \mathbf{x}_k^T\right) \\
\mathbf{Q} &= \frac{1}{K}\sum_{k=1}^{K}(\mathbf{y}_k - \mathbf{C}\mathbf{x}_k)(\mathbf{y}_k - \mathbf{C}\mathbf{x}_k)^T
\end{aligned} \tag{7}$$

3.3 Decoding arm trajectories from neural activity

After training the parameters, we compute $P(\mathbf{x}_k|\mathbf{y}_1, ..., \mathbf{y}_k)$ for $k = 1, ..., K$. $\{\mathbf{y}\}_1^k$ denote the sequence $\mathbf{y}_1, ..., \mathbf{y}_k$. The variables $\mathbf{x}_1, ..., \mathbf{x}_K, \mathbf{y}_1, ..., \mathbf{y}_K$ are jointly Gaussian, and so $P(\mathbf{x}_k|\{y\}_1^k)$ is also Gaussian. In the case where $P(\mathbf{x}_k|\{y\}_1^k)$ is Gaussian for all k, we only need to then infer its mean and covariance to fully describe the distribution. The Kalman filter is a way to compute $P(\mathbf{x}_k|\{y\}_1^k)$ recursively starting at $k = 1$. We do this by applying our state process (which propagates our prediction forward one time step) and then our observation process (which subsequently updates our distribution by using the most recently observed data). We call the former "one-step prediction" and the latter "measurement update".

One-step prediction:

$$P(\mathbf{x}_k|\{\mathbf{y}\}_1^{k-1}) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1})P(\mathbf{x}_{k-1}|\{\mathbf{y}\}_1^{k-1})d\mathbf{x}_{k-1} \tag{8}$$

Measurement update:

$$P(\mathbf{x}_k|\{\mathbf{y}\}_1^{k}) = \frac{P(\mathbf{y}_k|\mathbf{x}_k)P(\mathbf{x}_k|\{\mathbf{y}\}_1^{k-1})}{P(\mathbf{y}_k|\{\mathbf{y}\}_1^{k-1})} \tag{9}$$

To simplify expressions, we let

$$\begin{aligned}
\mu_{k|\tau} &= \mathbb{E}[\mathbf{x}_k|\{\mathbf{y}\}_1^{\tau}] \\
\Sigma_{k|\tau} &= \mathbf{cov}(\mathbf{x}_k|\{\mathbf{y}\}_1^{\tau})
\end{aligned} \tag{10}$$

One step prediction for jointly Gaussian models,

$$\begin{aligned}
\mu_{k|k-1} &= \mathbf{A}\mu_{k-1|k-1} \\
\Sigma_{k|k-1} &= \mathbf{A}\Sigma_{k-1|k-1}\mathbf{A}^T + \mathbf{W}
\end{aligned} \tag{11}$$

Now, the measurement update solves for $\mu_{k|k}$ and $\Sigma_{k|k}$. Letting $\Sigma_k = \Sigma_{k|k-1}$ and $(\tilde{\mathbf{x}})_k = \mu_{k|k}$, we arrive at the following recursions:

$$\begin{aligned}\tilde{\mathbf{x}}_k &= \mathbf{A}\tilde{\mathbf{x}}_{k-1} + \Sigma_k \mathbf{C}^T(\mathbf{C}\Sigma_K\mathbf{C}^T + \mathbf{Q})^{-1}(\mathbf{y_k} - \mathbf{C}\mathbf{A}\tilde{\mathbf{x}}_{k-1}) \\ \Sigma_k &= \mathbf{A}\Sigma_{k-1}\mathbf{A}^T + \mathbf{W} - \mathbf{A}\Sigma_{k-1}\mathbf{C}^T(\mathbf{C}\Sigma_{k-1}\mathbf{C}^T + \mathbf{Q})^{-1}\mathbf{C}\Sigma_{k-1}\mathbf{A}^T\end{aligned} \tag{12}$$

In above equation, the expression $(\mathbf{C}\Sigma_{k-1}\mathbf{C}^T + \mathbf{Q})^{-1}$ is called the *Kalman gain*, and is typically denoted by $\mathbf{K}_k$. The intuitive interpretation of this is that the next state, $\tilde{\mathbf{x}}_k$, is obtained by predicting one step forward via the state model $\mathbf{A}\tilde{\mathbf{x}}_{k-1}$, and then adjusting your step forward by adding $\mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\mathbf{A}\tilde{\mathbf{x}}_{k-1})$. The term $(\mathbf{y}_k - \mathbf{C}\mathbf{A}\tilde{\mathbf{x}}_{k-1})$ can be viewed as an error in how well the model predicted the next observation.

In summary, to test newly observed data, we set $\tilde{\mathbf{x}}_0$ = 0 and Σ_0 = 0. Then, we recursively calculate, until convergence:

1. $\Sigma_{k|k-1} = \mathbf{A}\Sigma_{k-1|k-1}\mathbf{A}^T + \mathbf{W}$
2. $\Sigma_{k|k} = \Sigma_{k|k-1} - \Sigma_{k|k-1}\mathbf{C}^T(\mathbf{C}\Sigma_{k|k-1}\mathbf{C}^T + \mathbf{Q})^{-1}\mathbf{C}\Sigma_{k|k-1}$
3. $\mathbf{K}_k = \Sigma_{k|k-1}\mathbf{C}^T(\mathbf{C}\Sigma_{k|k-1}\mathbf{C}^T + \mathbf{Q})^{-1}$

Now there are tow options to decoding. The first is to simply calculate $\tilde{\mathbf{x}}_k = \mathbf{A}\tilde{\mathbf{x}}_{k-1} + \mathbf{K}_k(\mathbf{y_k} - \mathbf{C}\mathbf{A}\tilde{\mathbf{x}}_{k-1})$ at each point in time along the recursion. However, we can also decode with the steady state form. After convergence of the recursion, we can compute $\mathbf{M}_1 = \mathbf{A} - \mathbf{K}_\infty\mathbf{C}\mathbf{A}$ and $\mathbf{M}_2 = \mathbf{K}_\infty$. Then we can decode the newly observed data via:

$$\tilde{\mathbf{x}}_k = \mathbf{M}_1\tilde{\mathbf{x}}_{k-1} + \mathbf{M}_2\mathbf{y}_k \tag{13}$$

The performance of these two different approaches should only differ early on (when $\mathbf{K}_k$ has not yet converged to $\mathbf{K}_\infty$), but will both eventually result in the same decoded trajectories.

3.4 Data set

The data set is taken from Prof. Jonathan Kao class on Neural Signal Processing and Machine Learning (ECE 243A) which is a data structure containing simultaneous reaching (kinematic) data and neural spiking data. The data structure is an array of dictionaries, with each dictionary in the array corresponding to one trial performed by Monkey J. In this data, Monkey J is performing a reaching task in which he acquires a center target, and then a peripheral target. After acquiring the peripheral target, he comes back to acquire the center target, and then acquires another peripheral target. This task is called a "center-out-and-back" task as the monkey continuously reaches from the center to a peripheral target, and then back to the center. 506 trails were performed by Monkey J on 9 unique targets. All trails resulted in success i.e., Monkey J acquired the target in all trials. The system which observed the monkey's kinematics (Polaris) sampled the kinematics at approximately 16.67 Hz. The data structure also contains the neural data recorded from electrode arrays implanted in Monkey J's primary motor cortex(M1). There are 96 electrodes and each one is measuring spiking activity at millisecond resolution. The unique targets, reach trajectories and spike raster at one of the electrodes are shown in Figure 4.

3.5 Implementing Kalman filter and Results

In this problem, velocity Kalman filter is used for decoding neural activity. The first 400 trials are used as training data and the remaining 106 trials are used as testing data. We are going to use an assumption made in [1] which is that the monkey sees cursor whenever it is updated and therefore has no uncertainty in its position. this assumption removes the uncertainty in the cursor's position. Figure 5 compares the decoded trajectory using Kalman filter with the actual trajectory on test data. The mean squared error in trajectory position per trial is reduced from 5315.91 mm^2 (Optimal Linear Estimator) to 3308.94 mm^2 (Wiener Filter) which is further improved to 1731.87 mm^2 (using Velocity Kalman filter).

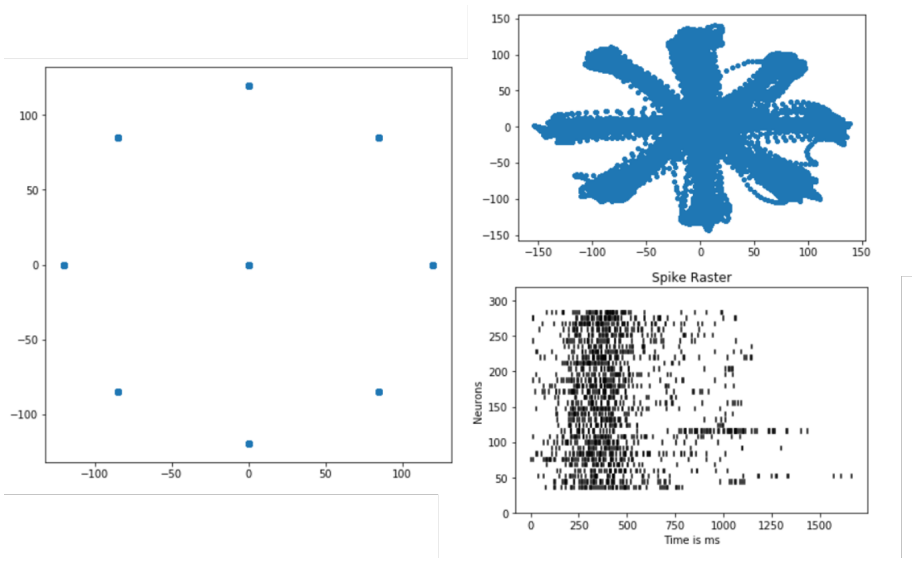

Figure 4: Plot of unique locations, Monkey J reach trajectories (for complete data) and spike raster at one of the electrodes

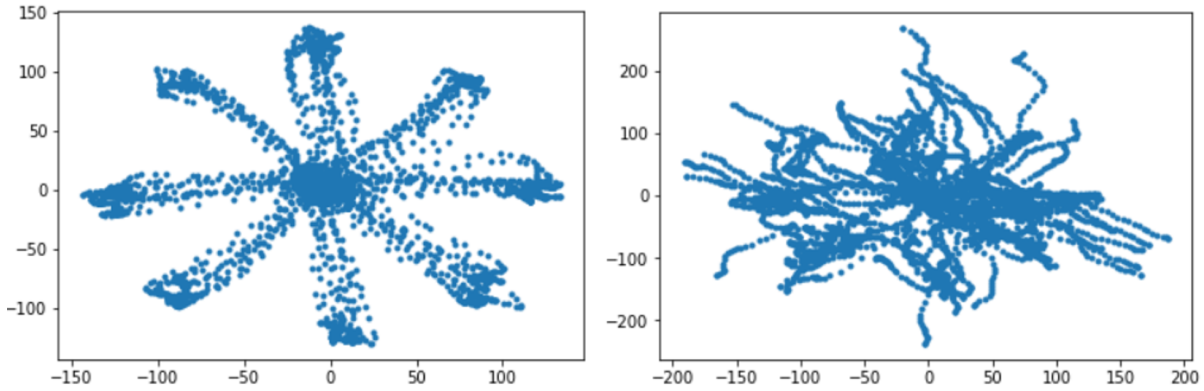

Figure 5: Plot comparing actual trajectory (left plot) and predicted trajectory through Velocity Kalman filter (right plot) on test data

4 Summary and Conclusion

This paper reviews the BMI systems and improves upon the previous decoding algorithms which are critical for their viability in clinical trials. Kalman filter improves on the previous decoding algorithms namely population vector and Wiener filter. Results shows that the MSE (mean squared error) in trajectory position per trail is approximately halved by using the Kalman filter as decoding algorithm. Prior to 2017, Kalman filter was state-of-the-art decoding algorithms but today, Recurrent Neural Networks showed superior performance than Kalman filter in decoding activity.

References

[1] Vikash Gilja, Paul Nuyujukian, Cindy A Chestek, John P Cunningham, M Yu Byron, Joline M Fan, Mark M Churchland, Matthew T Kaufman, Jonathan C Kao, Stephen I Ryu, et al. A high-performance neural prosthesis enabled by control algorithm design. *Nature neuroscience*, 15(12):1752, 2012.

[2] Thomas J Davidson, Fabian Kloosterman, and Matthew A Wilson. Hippocampal replay of extended experience. *Neuron*, 63(4):497–507, 2009.

[3] Christopher & Dana Reeve Foundation. *One Degree of Separation: Paralysis and Spinal Cord Injury in the United States*. Christopher & Dana Reeve Foundation, 2009.

[4] Kim D Anderson. Targeting recovery: priorities of the spinal cord-injured population. *Journal of neurotrauma*, 21(10):1371–1383, 2004.

[5] Jonathan C Kao, Sergey D Stavisky, David Sussillo, Paul Nuyujukian, and Krishna V Shenoy. Information systems opportunities in brain–machine interface decoders. *Proceedings of the IEEE*, 102(5):666–682, 2014.

[6] 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, 2008.

[7] 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.

[8] Dawn M Taylor, Stephen I Helms Tillery, and Andrew B Schwartz. Direct cortical control of 3d neuroprosthetic devices. *Science*, 296(5574):1829–1832, 2002.

[9] Thomas Kailath. A view of three decades of linear filtering theory. *IEEE Transactions on Information Theory*, 20(2):146–181, 1974.

[10] Norbert Wiener and Mass.) Massachusetts Institute of Technology (Cambridge. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Technology Press, 1950.

[11] Andrei Nikolaevitch Kolmogorov. Sur l'interpolation et extrapolation des suites stationnaires. *CR Acad. Sci*, 208:2043–2045, 1939.

[12] 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, 2006.

[13] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[14] Rudolf E Kalman. New methods in wiener filtering theory. In *Proceedings of the First Symposium on Engineering Applications of Random Function Theory and Probability, edited by JL Bogdanoff and F. Kozin, John Wiley & Sons, New York*, 1963.

[15] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[16] Wei Wu, Michael J Black, Yun Gao, M Serruya, A Shaikhouni, JP Donoghue, and Elie Bienenstock. Neural decoding of cursor motion using a kalman filter. In *Advances in neural information processing systems*, pages 133–140, 2003.