Monte Carlo Methods for Decoding Neural Spike Trains

Enrico Piperno

PHYSICS 212 - Stanford University

12/4/2024

1 Introduction

1.1 Motivation

The brain encodes information through neuron action potentials more commonly known as spikes. Spike sequences in neuron circuits encode information such as muscle movement, memory, and emotions. Decoding these spike patterns is a fundamental problem which can help those with neurological diseases. Brain-Computer Interfaces (BCIs) which enables communication with the human brain and external devices, has the possibility of revolutionizing the treatment of neurological diseases.

Parkinson's is a disease which affects movement and causes tremors. Current procedures involves medication and deep brain stimulation (DBS), which is a surgical procedure which involves planting electrodes into the brain to stimulate electrical activity. BCI, paired with EEG recordings on the skin, can monitor brain activity and stimulate the brain when necessary. BCI is adaptable to changing conditions, in contrast to DBS whose implanted electrodes cannot move inside the brain. Additionally, this example of BCI is non-invasive which eliminates side-effects from surgery (Fox, 2023). Another exciting application of neural decoding and BCI is to help paralysis patients. By understanding neural spike trains, $\vec{r}$, which result from a stimulus, $\vec{x}$, one can pair BCI with robotics to help regain movement.

1.2 Overview

In this project, we will begin by creating a generalized linear model (GLM), F, to model neural spike trains to stimulus (Pillow et al., 2008). The model will map the stimulus to spike trains as shown in Equation 1.

$$F : \{\vec{x}, stimulus\} \rightarrow \{\vec{r}, spike\,train\} \tag{1}$$

The model will include a cell's constant baseline firing rate and a stimulus receptive field. The model is not deterministic and thus models the stochastic attributes of neuron spiking.

After simulating spike trains to a stimulus, we will try to decode the model using two methods from Bayesian statistics: Maximum A Posteriori (MAP) and Bayesian Least Squares (BLS).

- MAP finds the mode of the posterior distribution, $p(\vec{x}|\vec{r})$, and is computed using gradient descent techniques. MAP provides the single most probable stimulus, $\vec{x}$, given a spike train, $\vec{r}$.
- BLS finds the mean of the posterior distribution and is computed using Markov Chain Monte Carlo (MCMC) with the Metropolis-Hastings (MH) algorithm. BLS provides a smoothed estimate of $\vec{x}$ given $\vec{r}$ which averages over the uncertainty in the stimulus-response mapping.

In this project, we will compare the efficiencies and accuracies of the two techniques. The code written for the project can be found on Github[1].

2 Model for Spike Trains

The ith neuron emits a spike train in response to $\vec{x}$ of the form,

[1] `https://github.com/EPiperno/Neural_Decoding`

$$r_i(t) = \sum_{\alpha} \delta(t - t_{i,\alpha}) \tag{2}$$

where $t_{i,\alpha}$ is the αth spike of the ith neuron. The collection of responses by all cells we define as $\vec{r}$.

The response, $\vec{r}$, of the cells is not completely resolved by the stimulus, $\vec{x}$, and is subject to trial-by-trial variations. We model $\vec{r}$ as a point process, i.e., events that are discrete and localized, by their instantaneous firing rate which comes from a Generalized Linear Model (GLM). This model is well established in literature (Chichilnisky, 2001), and is known as a Linear-Nonlinear-Poisson (LNP) model as shown in Figure 1.

Figure 1: Linear-Nonlinear-Poisson Model for Neural Response.

The LNP described below offers distinct advantages to modeling neuron response. First, the LNP has parameters with biological relevance such as a weight function on the stimulus and surrounding cell's inputs. Second, the LNP supports fitting to data to measure parameters. And lastly, the LNP calculates spikes using a Poisson distribution, thus capturing stochastic aspects of neuron firing beyond a determined function.

2.1 Linear

For every ith neuron, we create a real-valued linear function of the stimulus of the form:

$$u_i(t) = b_i + \int_0^{\tau} k_i(-t')\, \vec{x}(t - t')\, dt' \tag{3}$$

where b_i is the DC bias of the cell or the cell's baseline firing rate and k_i represents the cell's linear receptive field to the stimulus.[2]

[2]In the actual model (Ahmadian et al., 2011) surrounding cell's inhibitory or excitatory postspike effects were also included, however, we left these out for simplicity.

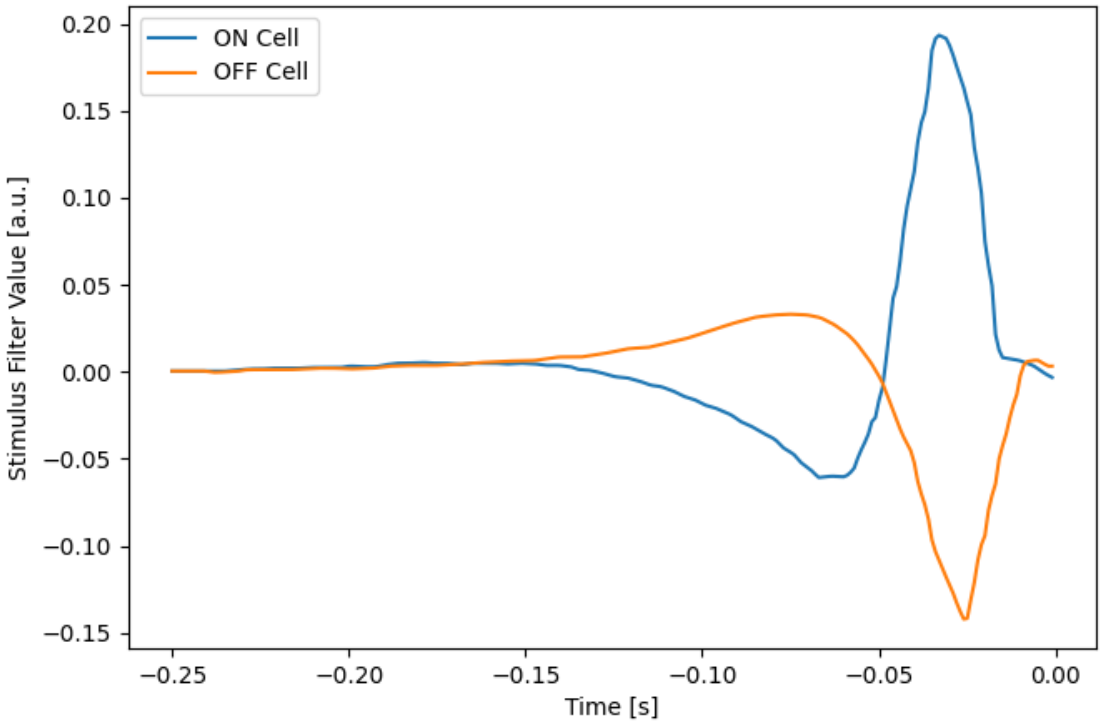

Figure 2: Receptive Field for ON and OFF Retinal Ganglion Cells.

In Figure 5, we show the example receptive fields for two Retinal Ganglion Cells (RGCs), taken from Pillow et al. (2011). Retinal ganglion cells are a type of neuron located in the retina which transmit visual information from the retina to the brain. They are two layers away from rods and cones, so they integrate visual signals and transmit them to the brain via their axons which form the optic nerve. ON RGCs increase their firing rate when light intensity in their receptive field *increases*, and OFF RGCs increase their firing rate when light intensity in their receptive field *decreases*. Notice how the receptive field is in negative time, such that it is applied on past stimulus and is thus is causal. Additionally, the peak for both ON and OFF cells is at -0.25s which implies there is a significant delay between stimulus and firing rate.

2.2 Nonlinear

Next, we introduce a nonlinear function to model the firing rate. In our case, we take:

$$\lambda_i(t) = e^{u_i(t)}. \tag{4}$$

We use a nonlinear function because neurons do not respond linearly to inputs, and a nonlinear function models their saturation or threshold behaviors. Additionally, it converts the linear function, $u_i(t)$, into a non-negative firing rate, suitable for the Poisson distribution governing spike generation.

2.3 Poisson

We use a Poisson distribution for explicity calculating the ith neuron's spike sequence, of the form:

$$P_i(N = n) = \frac{(\lambda_i(t)\Delta t)^n e^{-\lambda_i(t)\Delta t}}{n!} \tag{5}$$

where $\lambda_i(t)\Delta t$ is the expected number of spikes in Δt. In our model, we explicitly calculate whether a spike has occured by the following condition:

```
r_i[t] = np.random.rand() < (lambda_i[t]*dt)
```

where `np.random.rand()` returns a random number between $[0, 1)$.

2.4 Example Usage

In Figure 3, we show an example usage of the LNP model of ON and OFF RGCs. Notice how the log firing rate (proportional to $u_i(t)$) reacts to the stimulus for ON and OFF cells. That is, there is a significant delay for log firing rate with respect to stimuli, and for brief strong stimuli the log firing rate is not sensitive.

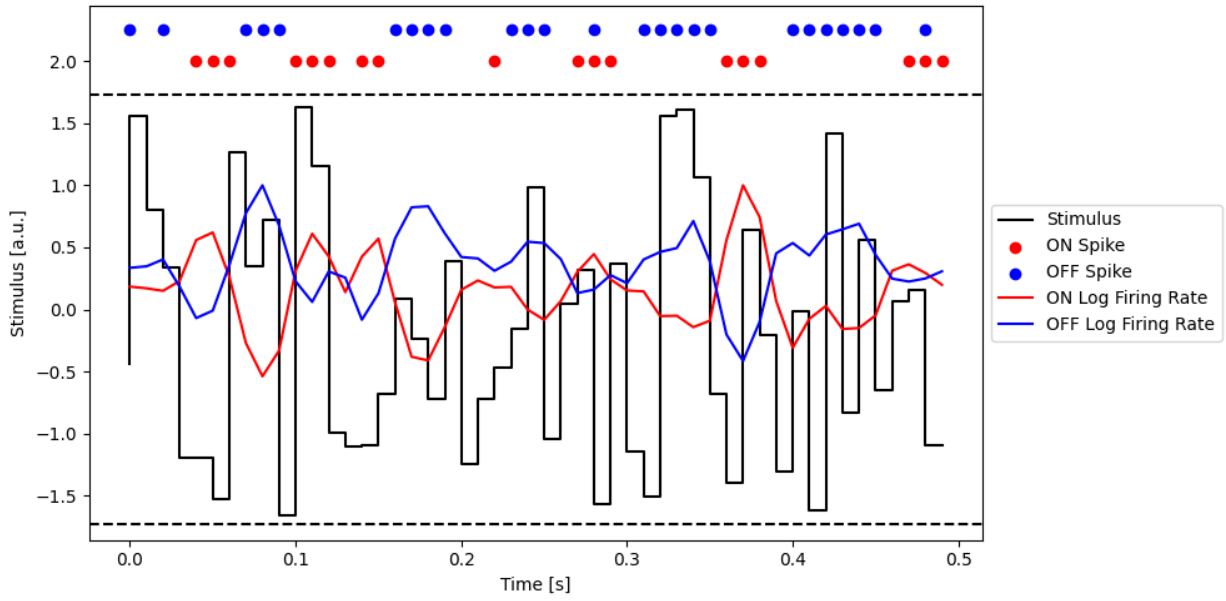

Figure 3: Example usage of Linear-Nonlinear-Poisson Neural Model for ON and OFF Retinal Ganglion Cells.

3 Maximum A Posteriori Decoding Technique

The MAP technique decodes the neural model by computing the mode or most likely stimulus given a spike train, $\vec{x}_{\text{MAP}}(\vec{r})$. This is done by maximizing the posterior probability distribution, $p(\vec{x}|\vec{r},\theta)$. However, this technique only considers a single point estimate of the decoded variable thus ignoring the spread of the posterior distribution which is important when the posterior distribution is multi-modal or skewed.

3.1 Algorithm Construction

Decoding the model is a matter of understanding the posterior probability distribution, which given by Bayes' rule is:

$$p(\vec{x}|\vec{r},\theta) = \frac{p(\vec{r}|\vec{x},\theta)p(\vec{x})}{p(\vec{r}|\theta)}, \tag{6}$$

where $\theta = \{b_i, k_i\}$ is the set of GLM parameters and where,

$$p(\vec{r}|\theta) = \int p(\vec{r}|\vec{x},\theta)p(\vec{x})\,d\vec{x},^{3} \tag{7}$$

is simply a normalization constant. The MAP estimate is by definition,

$$\vec{x}_{\text{MAP}}(\vec{r}) = \arg\max_{\vec{x}} p(\vec{x} \mid \vec{r}) = \arg\max_{\vec{x}} \big[\log p(\vec{r} \mid \vec{x}) + \log p(\vec{x})\big]. \tag{8}$$

For simplicity, we take the log of the posterior distribution, $p(\vec{x}|\vec{r})$, such that we can compute the arg max of the sum of the log of the forward probability, $p(\vec{r}|\vec{x})$, and the log of the prior distribution, $p(\vec{x})$.[4] We eliminate the log of $p(\vec{r})$ from the equation since it is independent of $\vec{x}$.

The forward distribution, $p(\vec{r}|\vec{x})$, can be written as (Snyder and Miller, 1991):

$$\log p(\vec{r} \mid \vec{x}) = \sum_{i,\alpha} \log \lambda_i(t_{i,\alpha}) - \sum_i \int_0^T \lambda_i(t)\,dt + \text{const.} \tag{9}$$

Since we take the stimulus, $\vec{x}$, to be uniformly distributed white noise with range $[-\sqrt{3}, \sqrt{3}]$[5]. $\log p(\vec{x})$ can also be taken out from the arg max expression. Leaving us with:

$$\vec{x}_{\text{MAP}}(\vec{r}) = \arg\max_{\vec{x}} \big[\sum_{i,\alpha} \log \lambda_i(t_{i,\alpha}) - \sum_i \int_0^T \lambda_i(t)\,dt\big]. \tag{10}$$

We use the python function,

`scipy.optimize.minimize()`,

to compute $\vec{x}_{\text{MAP}}$. This function finds the local minimum (or maximum) of a scalar function using gradient descent techniques.

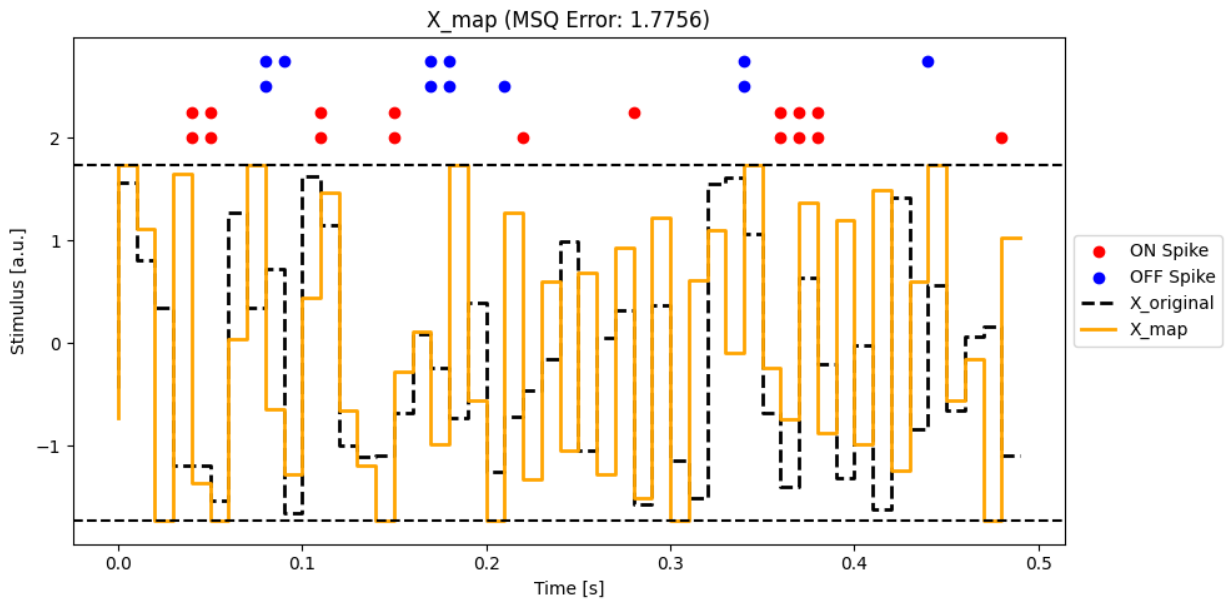

Figure 4: Example Maximum A Posteriori for decoding 4 ON and OFF RGCs. $\vec{x}_{\text{original}}$ represents the original stimulus, while $\vec{x}_{\text{MAP}}$ represents the predicted stimulus.

3.2 Example usage

As shown in Figure 3.2, we show an example MAP estimate for neural decoding. The Mean Squared Error for the predicted stimulus is 1.778.

4 Bayesian Least Squares (MCMC) Decoding Technique

The BLS technique decodes the neural model by computing the mean of a stimulus given a spike train, $\vec{x}_{\text{BLS}}(\vec{r})$. This is done by computing the posterior mean, given by Equation 11. This technique considers the spread and uncertainty of the distribution. This is important when the data is noisy or stochastic as is the nature of spike trains.

[3]For clarity we will drop θ from further equations.
[4]This holds because of the log-concave nature of the distributions.
[5]Such that the mean and variance is 0 and 1, respectively.

4.1 Posterior Mean Construction

The posterior mean is,

$$\vec{x}_{\text{BLS}}(\vec{r}) = E(\vec{x}|\vec{r}) = \int \vec{x}\, p(\vec{x}|\vec{r})\, d\vec{x}, \tag{11}$$

which is the optimal estimator which minimizes the mean squared error (MSE),

$$\text{MSE} = \frac{1}{n}\sum_{i}^{n}(\vec{x}_i - \hat{\vec{x}}_i)^2, \tag{12}$$

where $\vec{x}$ is the original stimulus and $\hat{\vec{x}}$ is the predicted stimulus.

Since calculating the posterior mean requires calculating a high-dimensional integral (Eq. 11), we can use the Monte Carlo method for approximating this integral. The Monte Carlo method which computes the integral by random sampling the probability distribution, $\pi(\vec{x})$, is,

$$\hat{\vec{x}}_N^{(\pi)} = \frac{1}{N}\sum_{t=1}^{N}\vec{x}_t, \tag{13}$$

where $\vec{x}_t$ are N independent and identically distributed (i.i.d.) samples from the distribution $\pi(\vec{x})$, and $\hat{\vec{x}}_N^{(\pi)}$ is the Monte Carlo estimate of the integral. By the law of large numbers, the Monte Carlo estimate approaches the true value of the integral as $N \to \infty$:

$$\hat{\vec{x}}_N^{(\pi)} \to \int \vec{x}\, \pi(\vec{x})\, d\vec{x}, \tag{14}$$

where ideally $\pi(\vec{x}) = p(\vec{x}|\vec{r})$. The strength of the Monte Carlo method is that it is not necessary to know the normalization constant, $p(\vec{r})$, of the full probability distribution, $p(\vec{x}|\vec{r})$. Additionally, since $p(\vec{x})$ is a uniform distribution as discussed in Section 3, we thus take $\pi(\vec{x}) = \log p(\vec{r} \mid \vec{x})$ as defined in Equation 9.

4.2 Markov Chain Construction

To find a computational efficient set of samples, $\vec{x}_t$, we compute an ergodic chain using the Metropolis-Hastings (MH) algorithm. An ergodic chain is a Markov chain[6] which efficiently explores a state space and whose long-term averages converges to the expected value of the target distribution.

We start with an initial value, $\vec{x}_0$, for the Markov chain. Then, we choose a new proposal distribution, $q(\vec{x}', \vec{x}_t)$ which proposes a new state, $\vec{x}'$, given the current state, $\vec{x}_t$. In our case, we take the proposal distribution to be:

$$\vec{x}' = \text{clip}(\vec{x}_t + \epsilon, -\sqrt{3}, \sqrt{3}) \tag{15}$$

where, [7]

$$\epsilon \sim \mathcal{N}(0, 0.1^2). \tag{16}$$

Next, we compute the acceptance probability,

$$\alpha = \min\left(1, \frac{\pi(\vec{x}_t)q(\vec{x}'|\vec{x}_t)}{\pi(\vec{x}')q(\vec{x}_t|\vec{x}')}\right). \tag{17}$$

Taking a random number, $u \sim \text{Uniform}(0, 1)$, we accept the proposed state, $\vec{x}'$, if $u \leq \alpha$ thus setting $\vec{x}_{t+1} = \vec{x}'$. Otherwise, we keep the current state $\vec{x}_{t+1} = \vec{x}_t$.

In simple terms, the ergodic chain tests whether the new state is more probable in the target distribution, $\pi(\vec{x})$, and always takes the new state if it is more probable than the old state and randomly takes the new state if it is less probable than the old state. This allows for the target distribution to be efficiently navigated and lets the long-term average of the ergodic chain to converge to the expected value of the target distribution.

[6] A process is Markov if its future state depends only on the current state and not the history of states, i.e., it is memoryless. $P(X_{t+1}|X_t, X_{t-1}, ..., X_0) = P(X_{t+1}|X_t)$.

[7] $\text{clip}(x, min, max)$ is done so that the state does not go out of bounds.

4.3 Example Usage

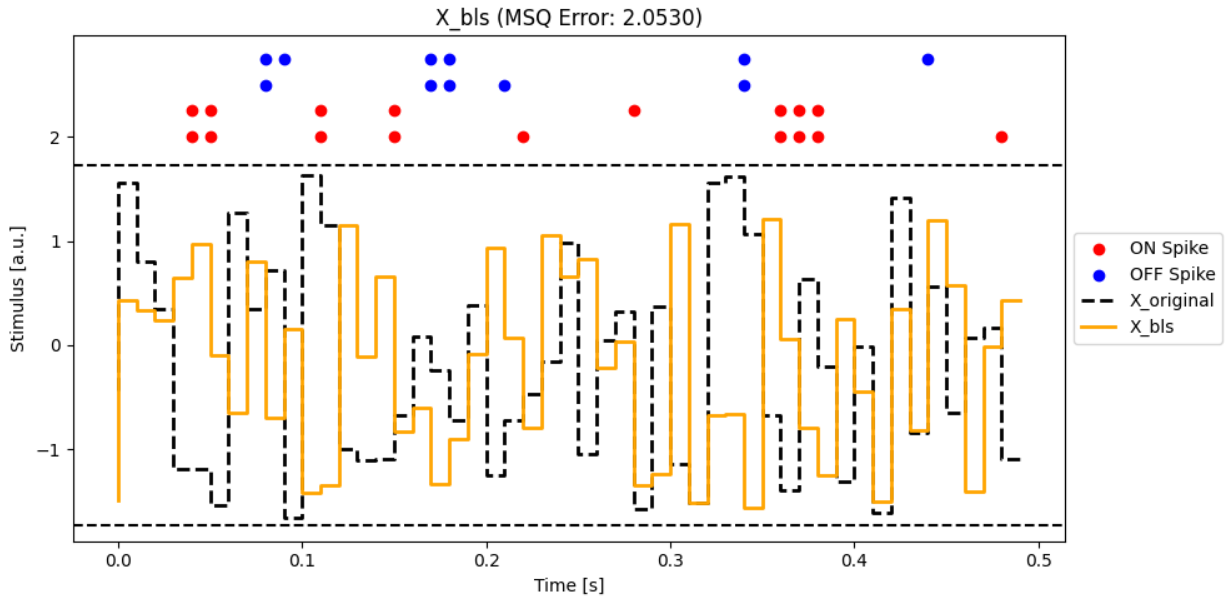

Figure 5: Example Bayesian Least Square Estimate (Posterior Mean) for decoding 4 ON and OFF RGCs. $\vec{x}_{\text{original}}$ represents the original stimulus, while $\vec{x}_{\text{BLS}}$ represents the predicted stimulus.

As shown in Figure 4.3, we show an example BLS estimate for neural decoding. The ergodic chain for computing the posterior mean was computed using the Metropolis-Hastings algorithm with 450 samples (after burning the first 50 samples). The Mean Squared Error for the predicted stimulus is 2.053.

5 Discussion

In the project, we discuss two techniques for decoding a stimulus, $\vec{x}$, from given spike trains, $\vec{r}$. The two techniques from Bayesian statistics are computing the mode and mean of the posterior distribution, $p(\vec{x}|\vec{r})$. To compute the mode or $\vec{x}_{\text{MAP}}$, we minimize a cost function. While to compute the mean or $\vec{x}_{\text{BLS}}$, we use a Monte Carlo technique with a Metropolis-Hastings Markov chain.

5.1 Analysis Results

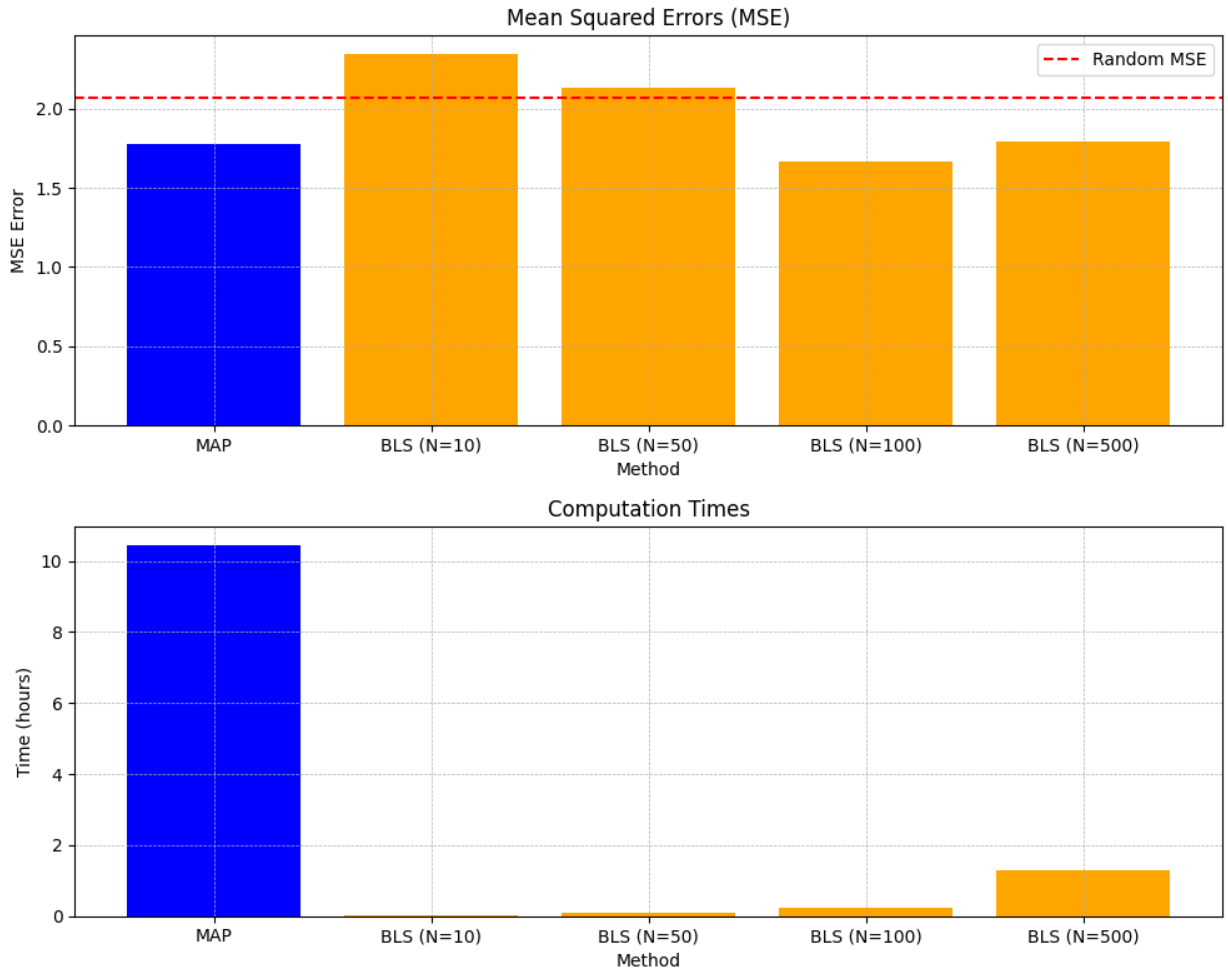

Figure 6: Accuracy and computational times of decoding. We present the MSE (Equation 12) as metric for accuracy and the computation time for the two decoding techniques as a measure of efficiency. N represents the length of the ergodic chain in the BLS technique, of which we burn the first 10%.

As shown in Figure 5.1, the accuracy of the two decoding techniques is very poor. We believe the accuracy is a result of the limited number of modeled neurons and stimulus filter weight[8]. Given that for a random predicted stimulus the expected value of the MSE ~ 2, the MSE of most of the estimates are worse than the average random predicted stimulus.

[8]Procrastination of the writer was another hidden factor.

5.2 Limitations and Future Work

As Figure 5.1 suggests, decoding neural populations is a challenging task. The first reason being that decoding requires solving high dimensional integrals (Equations 8 and 11) of which the computation grows exponentially with the number of dimensions[9]. Second, since neural activity is fundamentally stochastic[10], decoding becomes a challenge when the encoding function is not deterministic.

Future work could entail testing more model parameters. For instance, it would be interesting to increase the number of modeled neurons, decreasing the time steps, and adjusting the cell's baseline firing rate and stimulus filter weight. I suspect that increasing the stimulus filter weight and increasing the number of modeled neurons would increase the convergence of the MAP and BLS estimates as shown in Figure 6 of Ahmadian et al. (2011).

In conclusion, decoding neural populations remains an interesting and complex problem with large benefits. Decoding combined with Brain Computer Interfaces is the future of neural health from those suffering with neurological diseases such as paralysis, parkinson's, and more.

[9] Especially because they were run on local PCs without any parallelization. Longer runs were computed on a lab PC.

[10] For instance the cell's baseline firing rate, represented by b_i in the LNP model, is an example of the stochastic nature.

References

Ahmadian, Y., J. W. Pillow, and L. Paninski, 2011: Efficient markov chain monte carlo methods for decoding neural spike trains. *Neural Comput.*, **23(1)**, 46–96.

Chichilnisky, E. J., 2001: A simple white noise analysis of neuronal light responses. *Network (Bristol, England)*, **12(2)**, 199–213.

Fox, S., 2023: Brain-computer interfaces for the treatment of neurological disorders. Accessed: 2024-11-26.

Pillow, J. W., Y. Ahmadian, and L. Paninski, 2011: Model-based decoding, information estimation, and change-point detection techniques for multineuron spike trains. *Neural Computation*, **23(1)**, 1–45.

Pillow, J. W., J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. J. Chichilnisky, and E. P. Simoncelli, 2008: Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, **454(7207)**, 995–999.

Snyder, D. and M. Miller, 1991: *Random Point Processes in Time and Space*. Springer-Verlag, Berlin.