

Neural Signal Processing Tutorial II: Point Process Model Estimation and Goodness-of-Fit Analysis

Uri T. Eden, PhD¹, Lakshminarayan Srinivasan, PhD²,
and Sridevi V. Sarma, PhD³

¹Department of Mathematics and Statistics, Boston University
Boston, Massachusetts

²Department of Neurosurgery, Massachusetts General Hospital
Boston, Massachusetts

³Department of Brain and Cognitive Systems, MIT
Cambridge, Massachusetts

Introduction

Statistical models are used to explicitly propose a relationship between neural spiking activity and other measured signals, including biological stimuli and behavioral covariates. Generally speaking, these models can be useful for several tasks: summarizing spike train data, describing structure or patterns in the data, making inferences about the neural system, and estimating signals or dynamic properties of the firing activity. Neural models of spike train activity can also include biological models (e.g., Hodgkin-Huxley-type neurons or networks) that more concretely depict the mechanisms by which these relationships arise.

This tutorial goes through the basic steps for constructing and evaluating a statistical model for a spiking neuron. In particular, we will use generalized linear models (GLMs) to construct an inhomogeneous Poisson model of a place cell in the CA1 region of rat hippocampus. The GLM framework provides a flexible class of conditional intensity models that are easily fit using maximum likelihood methods and can be easily implemented in many standard mathematical and statistical packages, such as Matlab.

As we perform this analysis, here are some focus questions to keep in mind: First, what constitutes a statistical model for spike train data? Second, how do we specify a particular statistical model for a particular neural system? Finally, once a model class is specified, how can we choose the model parameters that are most in line with an observed spike train? We can answer the first question immediately: The statistical model specifies the probability of observing some number of spikes in any time interval, given (i.e., conditioned on) related covariates such as stimulus values or the spiking history. We will explore the answers to the other questions below.

Experimental Setup

In this tutorial, we will analyze spike train data recorded from a CA1 hippocampal neuron in a Long-Evans rat freely foraging in an open circular environment (70 cm in diameter with walls 30 cm high and a fixed visual cue). Both place-cell microelectrode array recordings and position data were captured from the animal. The neural activity was sampled at 1000 Hz, and the rat's position was sampled at 30 Hz. These experimental methods have been previously reported in detail (Brown et al., 1998).

Constructing a statistical model

The first step in performing statistical analysis of this spiking activity is to visualize the data in a way that

clarifies the relationship between the spiking activity and the rat's position.

Plotting raw data

To plot the raw data, we must first load in the data from a Matlab-readable file (e.g., .csv, .txt, .mat) using the “load” command. We can see what variables lie in the data set that we loaded by typing the command “who,” as shown below. You may type “help load” to see how to load files of different formats.

```
>> load glm_data.mat
>> who
```

Your variables are listed in Table 1.

Table 1. Variables contained in glmdata.mat

spike_times	Time stamps on spike events
x_at_spike_times	Animal position (x) at corresponding times stamps
y_at_spike_times	Animal position (y) at corresponding times stamps
T	Time
spikes_binned	spike data in 33 ms bins
xN	Normalized position (x)
yN	Normalized position (y)
vxN	Normalized velocity (v_x)
vyN	Normalized velocity (v_y)
R	Movement speed = $\sqrt{v_x^2 + v_y^2}$
phi	Movement direction = $\text{atan2}(v_y, v_x)$

Use Matlab to visualize the spiking activity as a function of time and then as a function of the animal's position.

Using the “plot” command, you should be able to generate the plot shown in Figure 1, which shows the rat's trajectory (blue) over 23 minutes and the spikes (red) for the single neuron. Notice how this particular neuron fires more when the rat is in the bottom, slightly left section of the circular environment.

Choosing a model form

The model is a function that specifies the probability of observing ΔN_k spikes in the k th interval of length Δ milliseconds. It is premised on parameters of the model (θ) and values of other signals of interest, i.e., covariates ($x^{(1)}, x^{(2)}, \dots, x^{(N)}$), that are postulated to affect the current probability of spiking, such as our stimulus and the occurrence of previous spikes:

$$\Pr(\Delta N_k | \theta, x^{(1)}, \dots, x^{(N)}) = f(\Delta N_k, \theta, x^{(1)}, \dots, x^{(N)})$$

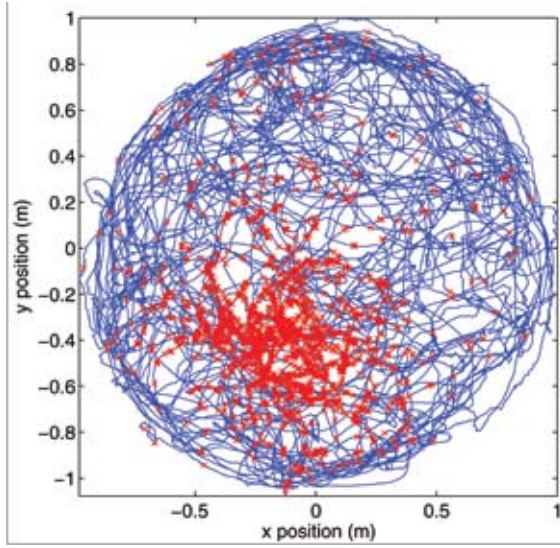


Figure 1. Rat's trajectory (blue) and spikes (red) for the single neuron firing.

Let's explore one way to specify this probability. First, define the function λ_k that can be used to calculate the instantaneous probability that a spike will occur. We can introduce parameters and covariates into the model by defining λ_k in terms of θ and $x^{(1)}, \dots, x^{(N)}$.

For this exercise, we will assume that the bin size is set small enough that the covariates are approximately constant within the bin, so that λ_k will be approximately constant over each bin. We assume the number of spikes that arrive in the k th bin is distributed as a Poisson distribution with mean and variance $\lambda_k \Delta$.

To make parameter estimation easier, we restrict the relationship between λ_k , θ , and $x^{(i)}$ to follow the GLM framework for a Poisson distribution (Brillinger et al., 1988; Truccolo et al., 2005). This means simply the following:

$$\log(\lambda_k) = \beta_o + \sum_{j=1}^J \alpha_j f_j(x(k))$$

where $\theta = \{\beta_o, \alpha_1, \alpha_2, \dots, \alpha_j\}$, $x(k) = \{x^1(k), x^2(k), \dots, x^N(k)\}$ and $\{f_1, f_2, \dots, f_j\}$ is a set of general functions of the covariates. Note that the log of λ_k is linear in θ , and that the functions can be arbitrary, nonlinear functions. This makes the GLM model class quite broad. In addition, if different types of covariates (e.g., extrinsic vs intrinsic) independently impact the

firing probability, then the GLM can model these separately as follows:

$$\log(\lambda_k) = \beta_o + \sum_{j=1}^J \alpha_j f_j(\text{extrinsic}(k)) + \sum_{m=1}^M \gamma_m g_m(\text{intrinsic}(k))$$

The GLM is an extension of the multiple linear regression model in which the variable being predicted (in this case, spike times) need not be Gaussian (McCullagh and Nelder, 1989). GLM provides an efficient computational scheme for model parameter estimation and a likelihood framework in which to conduct statistical inferences based on the estimated model (Brown et al., 2003).

✎ Pick a set of covariates (remember, this can include any function of the variables above), and write an equation that linearly relates $\log(\lambda_k)$ to your covariates.

Writing down the data likelihood

The data likelihood is simply the probability of observing the specific sequence of spike counts, conditioned on selected values of the parameters, and the actual values of the other covariates: $L(\theta) = \Pr(\Delta N_1, \dots, \Delta N_k | x^{(1)}, \dots, x^{(N)}, \theta)$, where the observed data are held fixed.

✎ Write down an equation for $L(\theta)$.

In order to derive the likelihood function, we consider a case where the time bin Δ is so small that only 0 or 1 spike can occur (e.g., $\Delta = 1$ ms). The spike train then forms a sequence of conditionally independent Bernoulli trials, with the probability of a spike in the k th time interval given by $\Pr(\text{spike in } (k\Delta, (k+1)\Delta] | x^{(1)}, \dots, x^{(N)}, \theta) \approx \lambda_k \Delta$. This yields the following joint probability:

$$\Pr(\Delta N_1, \dots, \Delta N_k | x^{(1)}, \dots, x^{(N)}, \theta) \approx \prod_{i=1}^k [\lambda_i \Delta]^{\Delta N_i} [1 - \lambda_i \Delta]^{1 - \Delta N_i}$$

For small Δ , we get the result that $[1 - \lambda_j \Delta] \approx e^{-\lambda_j \Delta}$ and $\log([\lambda_i \Delta][1 - \lambda_i \Delta]^{-1}) \approx \log(\lambda_i \Delta)$; therefore,

$$\Pr(\Delta N_1, \dots, \Delta N_k | x^{(1)}, \dots, x^{(N)}, \theta) \approx \prod_{i=1}^k \left[\frac{\lambda_i \Delta}{1 - \lambda_i \Delta} \right]^{\Delta N_i} [e^{-\lambda_i \Delta}]$$

Then, taking the log of both sides results in the following data likelihood function:

$$L(\text{Spike Train} | \theta) = \exp\left(\sum_{k=1}^T \log(\lambda_k \Delta t) \Delta N_k - \lambda_k \Delta t\right)$$

More details on deriving the data likelihood function can be found in Brown et al., 2003.

Choosing model parameters to maximize the data likelihood

We would now like to choose model parameters that most likely would have generated the observed data. This is accomplished by choosing the parameter estimate $\hat{\theta}$ that maximizes the data likelihood:

$$\hat{\theta} = \arg \max L(\theta)$$

Our problem of estimating model parameters has become a question of optimizing a function $L(\theta)$ over a possibly unconstrained space θ . Several methods can be used to approach optimization. Because we chose models built on the GLM framework, this optimization is computationally simplified (McCullagh and Nelder, 1989).

✎ Use the Matlab function `glmfit` to find the parameters of your model that maximize $L(\theta)$.

```
>> b = glmfit([covariate1 covariate2 ...], spikes_binned, 'poisson');
```

For small-parameter vectors, it is often useful to plot $L(\theta)$ as a function of θ around the parameter values returned by `glmfit` in order to confirm that `glmfit` indeed maximized the data likelihood. The shape of the likelihood around the maximum likelihood estimate determines the uncertainty about the estimates from the data.

Visualizing the model

✎ Using the equation for λ you wrote down in step 3, plot the intensity as a function of your covariates. Compare this graph with a plot of the signal values at which spikes occurred. This will visually confirm the plausibility of the model for explaining the relationship between the observed spiking and your covariates.

The script `glm_part1.m` is set up to fit and visualize a GLM for the spiking activity as a function of the rat's position.

```
>> glm_part1
```

This outputs a plot of the raw spiking data (Fig. 1) and another plot of the linear model of spiking rate (covariates = $[xN \ yN]$), both as a function of the animal's position (Fig. 2).

Notice that the linear model in Figure 2 cannot capture the phenomenon seen in the raw data (Fig. 1),

namely that in the lower, slightly lefthand side of the circle, the spiking decreases. Thus, a quadratic or Gaussian model may be more appropriate for capturing the place-field structure shown in Figure 1.

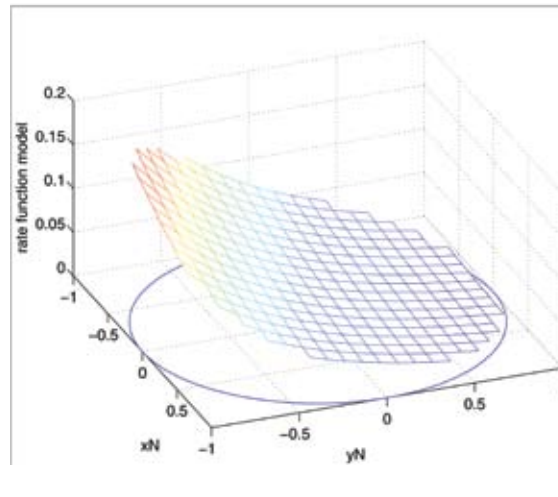


Figure 2. The linear model of spiking rate as a function of the animal's position.

✎ Modify the GLM in the script `glm_part1` to include other covariates and functions of covariates. Modify line 20 by including your new covariates in the model fit, and modify line 30 by writing in the equation for λ . Can you think of a GLM that can capture a Gaussian-shaped place-field structure?

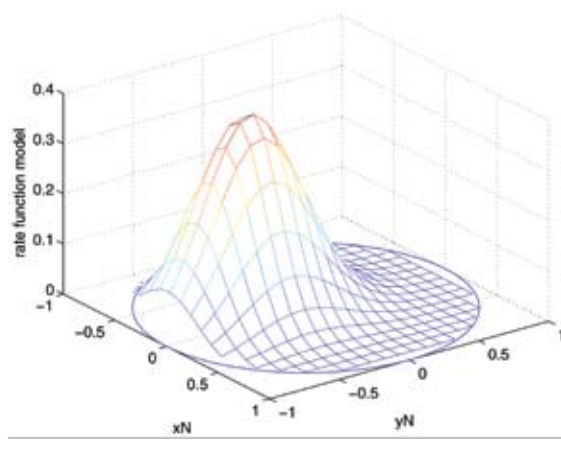


Figure 3. The Gaussian model of spiking rate as a function of the animal's position.

The script `glm_part2.m` is set up to visualize models of spiking as a function of velocity.

```
>> glm_part2
```

NOTES

Currently, this script outputs occupancy-normalized histograms of spiking simply as a function of the velocity covariates, as shown in Figure 4. Examining these histograms can provide insight into possible spiking models.

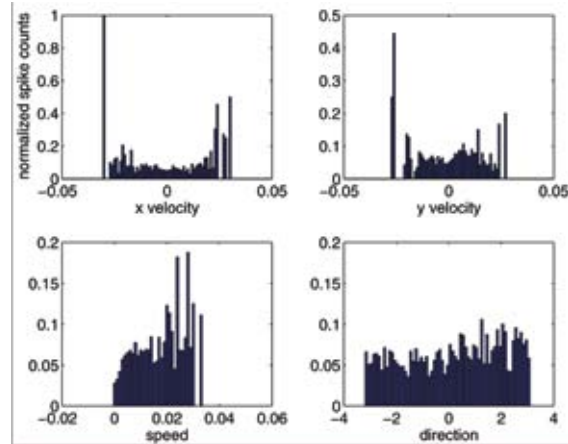


Figure 4. Occupancy-normalized histograms of spiking as a function of the velocity covariates.

✎ Modify the script `glm_part2` to fit GLM models to each of these variables, and then plot the GLM fits along with the occupancy-normalized histograms. What do these model fits suggest about the relationship between spiking and velocity?

Evaluating a statistical model

Now that we've written down a statistical model that explicitly relates neural signals with covariates such as stimulus values, and used maximum likelihood to choose values for parameters that were most consistent with the observed data, what comes next? In this section, we will investigate how to evaluate our model both in relative terms (how well it does compared with other models) and in absolute terms (how well it does in explaining the given data).

Here are some focus questions to consider as we attempt to evaluate the statistical models we just constructed: First, how precise are the model parameters that we just fit? Is it possible to simplify the model and still maintain accuracy? How do we know whether we have too many model parameters, and why might this be a bad thing? Can we quantitatively determine which of the models we constructed better fits the data and whether any of the models describes the data well enough to make useful inferences about the underlying properties of the neural system?

We will see that the data likelihood can also be used to determine precision in the choice of parameter, based on the Fisher information. We can use the data likelihood to provide confidence intervals for our parameters and eliminate parameters that are not distinguishable from zero. With too many parameters, a model may not generalize well to new data sets. The Akaike Information Criterion (AIC) and other measures quantify the tradeoff between fitting the data better and increasing the number of parameters. The Kolmogorov–Smirnov (KS) plot can help visualize how well the model explains the structure in all the data.

Determining the uncertainty of the parameter estimates

We can obtain confidence intervals around the parameter estimates based on the Fisher information. The observed Fisher information matrix is defined as the second partial derivative (the Hessian) of the log likelihood function $\log f(x|\theta)$ with respect to the individual parameters, evaluated at the following maximum likelihood estimate:

$$I_{obs}(\theta) = \frac{-\partial^2 \log f(x_{obs}|\theta)}{\partial \theta^2} \Big|_{\hat{\theta}_{ML}}$$

This quantity provides a sense for the precision of our model parameters. Recall that the ML estimate

of θ was chosen so that $\frac{\partial \log f(x_{obs}|\theta)}{\partial \theta} = 0$. If this

derivative gradually goes to zero at around θ_{ML} , then presumably, the neighboring values of θ_{ML} would have been almost as good and might have been chosen depending on the accuracy of the optimization procedure.

✎ Revisit the GLM models that you constructed in Part 1 of this tutorial. Load the data.

```
>> load glm_data.mat
```

Now call `glmfit()` in the following way to return additional statistics about the model fit:

```
[b,dev,stats] = glmfit ( arguments of the function call );
```

Generate a number of possible models with this command, including one with quadratic terms in x and y position in relation to the animal.

✎ Examine the confidence intervals computed for each parameter of two models based on the square root of the inverse of the observed Fisher information (σ) given in the variable `stats.se`. Use `errorbar()` to plot the parameters $b \pm 2\sigma$, and determine which parameters are statistically different from zero.

Figure 5, below, plots the confidence intervals for the quadratic model with six parameters (`[xN yN xN.^2 yN.^2 xN.*yN]`). Note that the sixth parameter has a confidence interval that includes 0; thus, the parameter is not statistically different from zero.

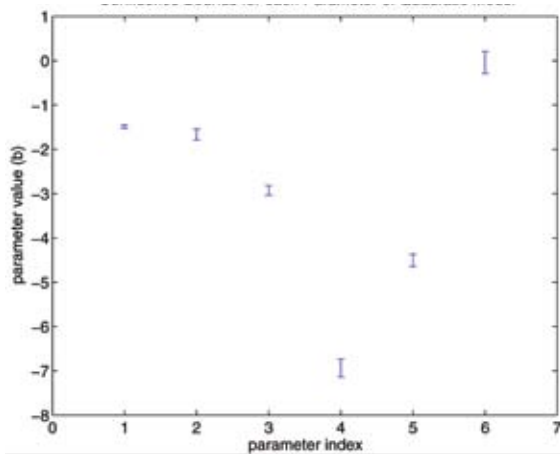


Figure 5. Confidence intervals for parameters of quadratic model.

✎ Examine the corresponding p value assigned to each parameter in `stats.p`. Which parameters have significant p values?

The corresponding p values for the six parameters in the quadratic model are all smaller than 10^{-4} except for the sixth parameter, which is 0.8740.

Constructing confidence intervals about your model conditional intensity

In the previous section, we computed the uncertainty about the parameter estimates. Now, we can propagate this uncertainty into the intensity model. This gives us a sense of how certain we can be about the conditional intensity as a function of the covariates.

✎ Use the Matlab function `glmval` to calculate confidence intervals about the conditional intensity.

```
>> [b, dev, stats] = glmfit([xN
xN.^2],spikes_binned,'poisson');
>> [yhat,dylo,dyhi] = glmval(b,[-1:.01:1;
[-1:.01:1].^2'],'log',stats);
>> errorbar([-1:.01:1],yhat,dylo,dyhi);
```

If you build the above model, you will get the uncertainty into the intensity model shown in Figure 6, below.

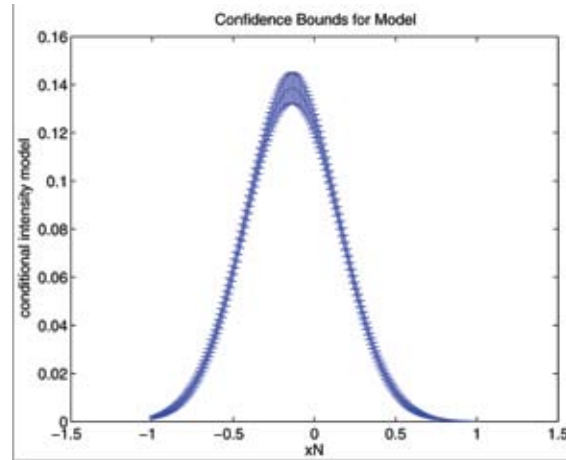


Figure 6. Uncertainty propagated into the intensity model for xN-quadratic model.

Characterizing the relative goodness-of-fit among competing models

Now we have a sense of which parameters contributed to the model in explaining the relationship between covariates and the observed neural activity. But how do we compare the quality of competing models, especially when they may have different numbers of parameters?

One answer is to examine the deviance of each model, defined as -2 times the log of the data likelihood:

$$\text{deviance} = -2\log f(x | \theta)$$

This quantity grows smaller as the data become more likely under a particular model. It is a generalization of the mean-squared error under a Gaussian linear model.

We would also like to penalize our error for having too many parameters. Intuitively, having too many parameters may restrict the ability to allow the model to generalize to (predict) new data. The AIC balances deviance against the number of parameters P :

$$\text{AIC} = -2\log f(x | \theta) + 2P$$

✎ Compare the AIC between two models. For example, consider the GLM models that are linear versus quadratic in relation to the animal's position.

NOTES

The AIC for the linear model where covariates include $[x_N y_N]$ is $1.6175e + 004$, while the AIC for the quadratic model $[x_N y_N x_N.^2 y_N.^2 x_N.*y_N]$ is $1.2545e + 004$. These criteria indicate that the quadratic model is considerably better than the linear model.

⌘ How can the AIC be improved for the GLM quadratic model of the animal's position?

We saw above that the cross-term between x_N and y_N has an associated parameter that is close to zero and therefore does not contribute significantly to the model. If we eliminate this parameter and build a new quadratic model with covariates $[x_N y_N x_N.^2 y_N.^2]$, we get an AIC of $1.2543e + 004$, which is slightly less than that of the previous quadratic model. This suggests that the model without the quadratic cross-term provides a more parsimonious description of the data.

⌘ How is the AIC of a given model affected by eliminating statistically significant parameters?

If we remove the first parameter of the linear model (x_N) and build a GLM model with covariate y_N , we get an AIC of $1.6315e + 004$, which is higher than that of the full linear model. In general, removing statistically significant parameters significantly increases the AIC.

Characterizing the goodness-of-fit between data and model

The AIC allowed us to compare alternative models. But how good is the best model in explaining the statistical structure in the data? The KS statistic provides one absolute measure.

Let's briefly revisit the time rescaling theorem. This theorem states that, with the correct conditional intensity function, interspike intervals (ISIs) can be transformed into an independent, identically distributed, exponential set of random variables. This transforms any point process into a unit rate Poisson process. Consequently, if we did have the correct conditional intensity function, the cumulative distribution on rescaled ISIs would match that of an exponential distribution with $\lambda = 1$.

To form the KS statistic, first rescale the empirical ISIs with your model conditional intensity function. Compare the cumulative distribution of these rescaled ISIs to that of an exponential distribution with $\lambda = 1$. The KS statistic is the maximum of the absolute value of the difference between these cumulative distributions.

⌘ Open `glm_part1_ks.m` and adjust the GLM models to your liking. Now scroll down to the KS Plot section of the code. Edit the code to correctly compute your model intensity at each point in time. Running the code will now output the KS statistic and plot, which includes the 95% confidence interval. Compare the KS statistic between two competing models. Which one does better? Does that model do a good job at explaining all the statistical structure in the data?

Figure 7 below shows the KS plots of the linear and full quadratic model with cross-term. Note that the quadratic model has better absolute goodness-of-fit to data. The fact that the KS plot for the quadratic model still does not fit entirely inside the 95% confidence intervals suggests that some statistical structure remains that this model fails to capture.

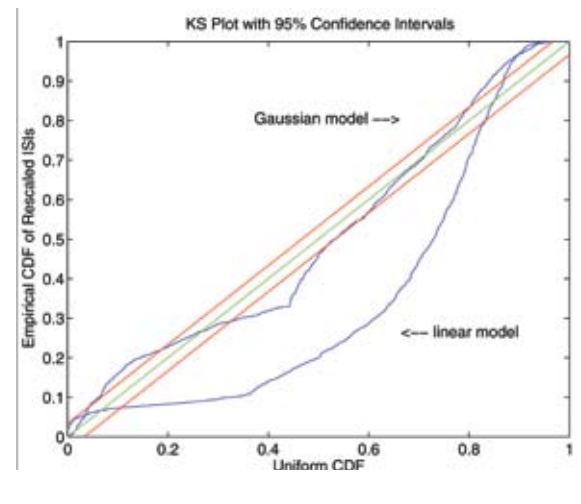


Figure 7. KS plots of linear and quadratic models.

⌘ How could this statistical model be further improved? What other variables might be related to the spiking activity of this neuron? Is there a single correct model for these data?

References

- Brillinger DR (1988) Maximum likelihood analysis of spike trains of interacting nerve cells. *Biol Cybern* 59:189-200.
- Brown EN, Frank LM, Tang D, Quirk MC, Wilson MA (1998) A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *J Neurosci* 18:7411-7425.

- Brown EN, Barbieri R, Eden UT, and Frank LM (2003) Likelihood methods for neural data analysis. In: Feng J, ed. Computational neuroscience: A comprehensive approach, Chap 9, pp 253-286. London: CRC Press.
- McCullagh P, Nelder JA (1989) Generalized linear models, 2nd ed. Boca Raton, FL: Chapman & Hall/CRC Press.
- Truccolo W, Eden UT, Fellow MR, Donoghue JP, Brown EN (2005). A point process framework for relating neural spiking activity for spiking history, neural ensemble and extrinsic covariate effects. J Neurophys 93:1074-1089.

NOTES