Fall 2018 EE 416 Final Project:

Professor James A. Ritcey
TA Daniel Zdeblick

Department of Electrical & Computer Engineering
University of Washington
Seattle, WA 98195

zdeblick@uw.edu jar7@uw.edu

November 30, 2018

Contents

1 EE 416 Project Rules

1. The Final Project for EE 416 is a group 2-person project. Each group should have two people in it. Both are to share equally in the work, and each will receive the same project grade, except for extraordinary circumstances. If you encounter group problems, talk to me or the TA. Report your group composition, your names, to Daniel.

2. I expect that each group does its own work, without collaboration with anyone. Cite any sources you use, including web pages. No plagiarism in any form is acceptable, and will have consequences. Plagiarism includes coding, copying, help from others beyond the TA and myself. You can use packages, but you must **cite** any packages that you use and show that you can use them correctly. Any language is acceptable. Typically Matlab, Python, or R will be used.

3. The report should be concise and complete. Due date and submission information is on our website. Please upload you codes separately and name you files by your group name, for example, RitceyZdeblick. Do not use EE416Project! We get a download folder with all the supporting files.

2 Data Download

For this project, we will be using data that was recorded from neurons in vitro (sliced out of a real brain and kept alive in a dish) and graciously uploaded for public use by the Allen Institute for Brain Science. The files you will need for this project are available on the Canvas Files page, under project/single_cell_data If you would like to check out the full dataset yourself (and all the other cool stuff the Allen has made available), go to http://celltypes.brain-map.org/ and browse around. Each number in the filename corresponds to a different cell that was recorded from independently; one file is a binary signal that is 1 when the neuron fired an action potential (spiked) in the corresponding time bin, and 0 otherwise; the other file is the stimulus signal, a current injection to the neuron (given in picoamps).

3 Introduction to Generalized Linear Models

Generalized Linear Models are models where the dependent variable is drawn from some distribution whose parameter is a nonlinear transformation of a linear function of the independent variables. In the case of a GLM model of neural spiking (which is often referred to as a Linear-Nonlinear-Poisson model or a Hawkes Point Process), the dependent variable is how many times a neuron spiked in a given time bin $n(t)$. The independent variables that this depends on are the stimulus and spike history from a finite number of previous time bins $s(t-dt), ...s(t-d*dt)$ and $n(t-dt), ...n(t-d*dt)$ respectively. Usually, the the distribution of spiking events is Poisson, and the nonlinear function is the exponential. Thus, we have

$$n(t) \sim Poisson(e^{f(0)s(t-dt)+..+f(d-1)s(t-d*dt)+h(0)n(t-dt)+..+h(d-1)n(t-d*dt)+b}),$$

where f and h are stimulus and spike history filters respectively, and b is a constant offset. A schematic for this model is shown in Figure 1. In our case, our stimulus will be a current injection, and not a visual stimulus, so our filters will just be temporal, not spatio-temporal.

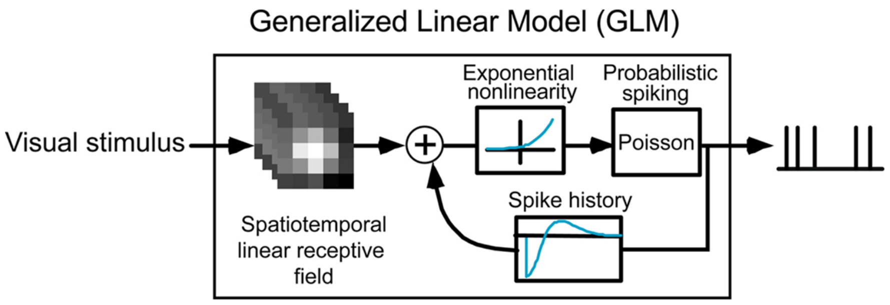

Figure 1:

4 Simulating and Fitting GLMs

4.1 Simulating a GLM

Write a function called 'sim_GLM', which should take as inputs a discretized stimulus and model parameters (stimulus filter, self-interaction filter, and offset), and output a sample spiking pattern of the neuron (a binary signal of the same length as the stimulus). For the purposes of this simulation, make sure that no bin has more than 1 spike - this will not affect your results much, it mostly just makes the data more similar to an actual neural recording.

Consider a model GLM neuron with offset $b = -15$, stimulus filter $f(t) = 20e^{-t}$, and self-interaction filter $h(t) = -200e^{-t}$. Truncate both filters to have a finite length of 15, and plot each of them in separate plots. Simulate this neuron's response to a white noise input with mean 0.3 and standard deviation 0.1, and plot both the stimulus and spiking response. Make the stimulus 20000 samples long, and set the first 2000 samples to 0. Your plots should look like Figure 2 (without the orange lines in the top two plots)

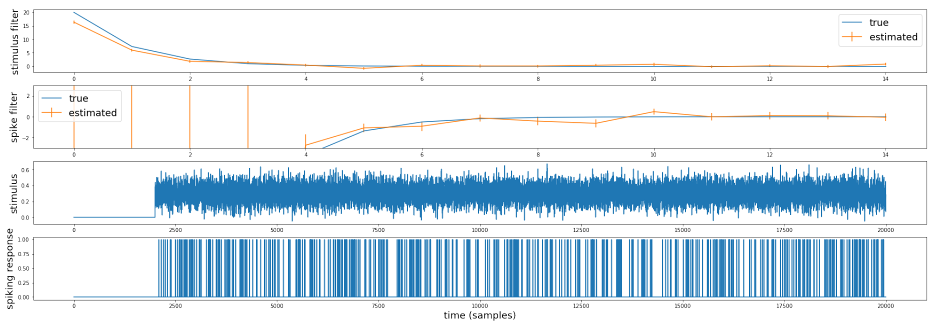

Figure 2:

4.2 Fitting Parameters to a GLM

Now that you can simulate a model GLM neuron, let's figure out how to fit the model parameters to data. In this section, you will write code to fit the parameters, test it on the model neuron you just simulated, and investigate how the amount of data affects the accuracy of the fit.

Write a function called 'fit_GLM' that takes as inputs a discretized stimulus and the spiking response of a neuron to that stimulus (a binary signal of the same length as the stimulus) and outputs the maximum likelihood model parameters (stimulus filter, self-interaction filter, and offset), along with the standard errors of these estimates.

To write this function, you should use a built in function to perform the optimization routine for a GLM - Matlab has the function '[b, ,stats] = glmfit(...)', and Python has the class 'GLM' in the statsmodels package, with the function model_results = GLM.fit(). For both of these, you must specify the distribution of the output variable (Poisson), but the nonlinearity defaults to the exponential (log for the link function). Also note that generally, the GLM functions take as input a matrix of the independent variables, where each variable is a column and each row is a time point, and a vector of the dependent variable. In this case, each independent variable is the stimulus or spike signal some fixed number of time steps before the time of the dependent variable (spiking). To get the standard errors of the parameter estimates, use model_results.bse in Python (don't ask what the b stands for), and stats.se in Matlab

Test this function on the stimulus and spikes you simulated, and plot the estimated filters (with standard errorbars) on top of their corresponding Ground Truths, as in Figure 2. Note that the spike filter has been zoomed in on.

Comment on how well this fit performed - are the estimated parameters accurate? In particular, comment on which particular parameters are inaccurate, and speculate on why this may be the case. Should we regard this as a failure in the fitting procedure? As such, how should we appropriately measure the error between the true and estimated filters using a Root-Mean-Squared-based or Mean-Standard-Error-based metric? (Hint: which filter coefficients should you use to compute the error?)

4.3 Understanding GLM Performance

Now write a script that repeats this process of simulating spiking and fitting model parameters for different lengths of stimulus, with several trials for each length. For each trial, compute the RMSE-based and Mean-Standard-Error-based errors that you came up with as well as the absolute error for the offset. Use stimulus lengths of 1000, 3000, 10000, and 30000 (setting the first 10th of the samples to 0), and 6 trials per length. Plot these errors as a function of the stimulus length - you should get something like the top row of Figure 3.

Comment on how the amount of data affects the accuracy of parameter fitting and on possible causes. What is the relationship between the standard errors and the mean squared errors? To answer this last question, it may be helpful to plot the two errors against one another.

Now, instead of focusing on the number of samples, let's focus on the number of spikes, since generally there are more time samples without spikes than with. For the previous analysis, re-plot the errors, now as a function of the total number of spikes in the data. This should look like the second row of Figure 3

Let's see if this is any different when the number of spikes is changing because of the neural dynamics instead of the amount of data. Fix the stimulus length to 10000 samples (again setting the first 10th of the samples to 0), and now vary the offset b, using values of -14, -16, and -18. Again, plot the errors against the total number of spikes. You should get something like the third row of Figure 3.

Comment on how the number of spikes in the data affects the accuracy of parameter fitting. Is this consistent with the effect of more data? Which is more relevant for fitting the model, having enough data or enough spikes? Comment on something else you learned in this section.

5 GLM fits of Allen Institute for Brain Science Data

Now we're ready to use GLM models to understand neurons that were recorded from by the Allen Institute. Fit GLM models with filters of length 20 to the data from both neurons, using the function you wrote in section 4.2. Plot both pairs of filters with error bars, and report the offsets and their errors.

Comment on the model fits: Which parameters were fit reliably and unreliably? Use what you learned last section about standard error! What can we infer from them about the response properties and dynamics of the neurons?

6 Inter-spike Interval Analysis

The spikes events can be thought of as random points in time, one following another. This is similar to the arrival of jobs at a processing center. One useful concept for random events in time is the inter-arrival time. In neuro, these are called the inter-spike intervals.

If the interspike intervals (call them the ISI) are

$$X_1,\ X_2\ldots,\ X_n,\ldots$$

The corresponding events times S_n, are

$$S_0 = 0,\ S_1 = X_1,\ S_2 = X_1 + X_2, \ldots$$

More generally,

$$S_n = \sum_{\ell=1}^{n} X_\ell$$

In general, the ISI are random, the only restriction is that the ISI X_n be positive, $X_n > 0$.

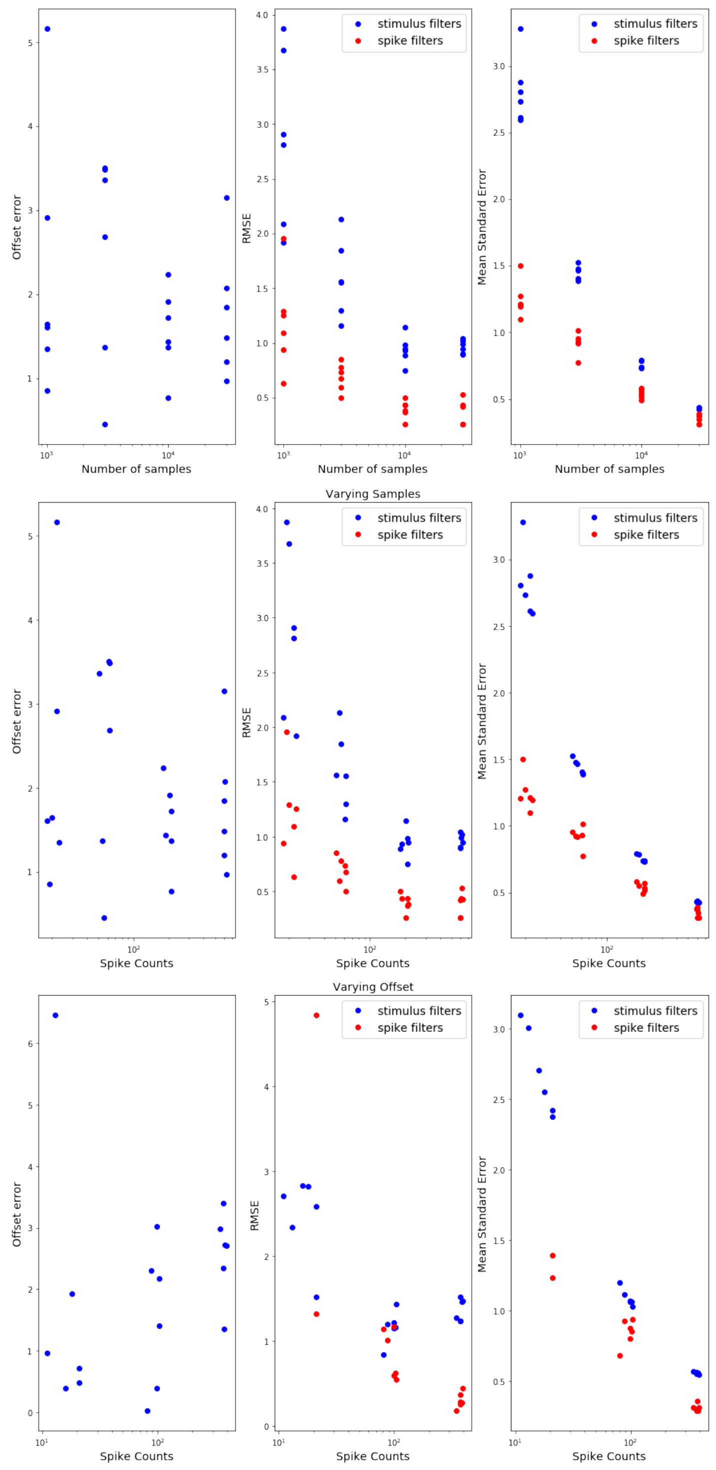
Offset error
RMSE
Mean Standard Error
stimulus filters
spike filters
Number of samples
Varying Samples
Spike Counts
Varying Offset

Figure 3:

The resulting sequence of events is called at discrete time point process, with the points being the S_n. Under various assumptions, we have special cases that are well-studied. For example, when the ISI are IID, independent and statistically independent, the resulting point process is called a renewal process. Renewal processes arise from reliability, in which the renewal times are the time of replacement for some failed equipment. The ISI are then the lifetimes of each replaced item. The most famous renewal process is the Poisson Process, for which the ISI have a common exponential distribution.

How many spikes occur in a time interval $(0, t]$ This is given by the *counting process*

$$N_t = N(t) \; = \; \#\{ \; S_n \text{ in } (0, t] \; \}$$

We can see that

$$\{ \; N(t) \leq r \; \}, \;\; \text{if and only if, } \{ \; S_n > t \; \}$$

and so (**corrected**)

$$P\{N_t \geq r\} = P\{S_r = \sum_{\ell=1}^{r} X_\ell \leq t\}$$

Thus, we can connect up the event times and counting process, with the ISI probability distribution.

Generally, the renewal hypothesis is not true, the ISI are not independent and their distribution can change. In the case of neural spiking, they will change due to the stimulus.

6.1 Detailed Tasks

Download the *simulated* dataset RenewalEventTimes.txt and analyze these to determine the ISI probability distribution. *This dataset is exactly a renewal process.*

1. Download the dataset RenewalEventTimes.txt
2. Compute the ISI from the event times.
3. Histogram and explore the dataset as you choose. Estimating the mean and variance for example, and estimating the correlation between successive ISI.
4. Suggest a probability density function that models the ISI, assuming the renewal model.
5. Overlay your model PDF with the histogram and provide a goodness of fit.

The neural spiking datasets are, of course, more complicated and most likely do not fit the renewal hypothesis. However, we can still learn something about the cells by looking at them. Histogram the ISIs of the second cell (the data in binned_spikes_cell_2.txt) and *comment on the appearance of the PDF*. Choosing a model for this data is *not required* for this section, but you must comment on the relationship of the *specific* ISI distribution to the *specific* GLM model parameters you discovered for this cell in section 5.

7 Detailed Tasks

1. EE 416 Project Rules
   - Read
2. Data Download
   - Download the datasets from Canvas EE416 Website https://canvas.uw.edu/courses/1219949/pages/2018-ee416-final-project
3. Introduction to Generalized Linear Models
   - Read and think about neurons - this will help when trying to analyze real data later
4. Simulating and Fitting GLMs

   4.1. Simulating a GLM
   - Write a function to simulate GLM models
   - Simulate a model with the given parameters and plot results. Describe

   4.2. Fitting Parameters to a GLM
   - Write a function that fits parameters to a GLM from data
   - Fit parameters to the simulated model's data and compare to the original parameters. Describe your results.
   - Adapt error metrics

   4.3. Understanding GLM Performance
   - Plot the two errors of both filters and offset error as a function of varying data length
   - Vary data length and plot the two errors of both filters and offset error as a function of number of spikes
   - Vary offset and plot the two errors of both filters and offset error as a function of number of spikes
   - Compare error metrics and analyze how amount of data/spikes affects accuracy
5. GLM fits of Allen Institute for Brain Science Data
   - Fit GLM models to neural data from 2 neurons
   - Analyze model parameters
6. ISI Analysis. The tasks were detailed in the section.

8 Reporting Format

Each two-person team must deliver the following group report by the due date. The goal is a concise report, figures should be referenced, and extraneous detail is to be eliminated. Grading is split among technical approach and results, report format and quality, and completeness. No report can exceed twenty pages; I would expect *much less* than this.

This handout is produced using a free online Latex editor called Overleaf, but it requires that you know Latex. It is excellent for typesetting math, but including figures is not as easy as with WORD. Writing Latex is more like programming, in that a compilation step is required.

The following format of the report is required.

1. *Title Page*, with your names, affiliation and title.
2. *Abstract*, a paragraph summarizing this problem and scope of your results.
3. *Summary of Results.* Your main *results*, without the clutter of supporting analysis. Include all graphs with detailed recommendations and answers to all questions posed in the problem statement in the order in which they are asked, labeled by section. Refer to figures in your text. Results should be given in the same order as the problem statement.
4. *Analysis.* Include any mathematical analysis and software flow upon which you base your design. You can hand-write this and scan it, if this is easier. This section should support your results.
5. *References.* Provide references to books, reports, class notes,websites, and papers where necessary. I have no problem with using outside sources that are acknowledged and cited.
6. *Appendices.* Place computer code in separate files. The code should have some structure and comments in it. Put figures in the earlier sections where they will be discussed.

9 Grading Rubric

We will upload a grading rubric. This will include points for correctness, completeness, and overall quality - readability. If you are seeking additional points, then you may add some original analysis, but please describe your methods and results. Thanks.