Generalized Linear Model (GLM)

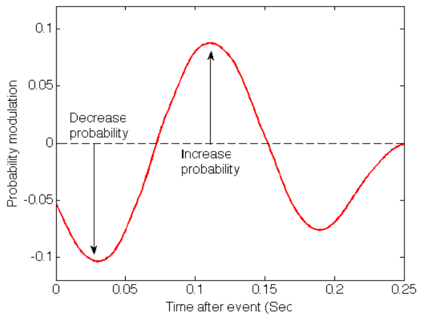
Yarden Cohen, Jul, 2013

Description:

This is a statistical framework that relates the instantaneous firing rate[1], $\lambda(t)$, of one cell to external stimuli and the spikes of other cells.

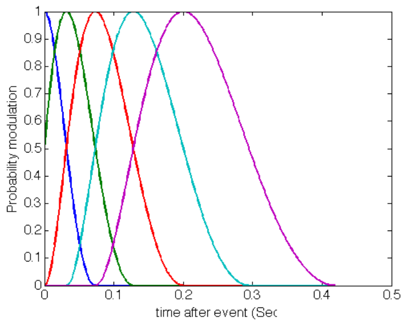
Each event (spike or stimulus) at time t_0 modulates the firing rate in its own way.

For example, an event at time 0 (X axis) in the graph below will modulate the spiking probability (Y axis, red line) by first decreasing it and than increasing it etc'.



The relation between an event and its influence on the spiking probability is called 'a filter'. In general, these filters can take any form but in order to use fewer parameters we choose a family of 'bumps' to span all filters[2].

The following graph describes the basis of filters:



Each colored line descibes a filter in the basis. The basis is defined by the distribution of the peaks and the widths of the bumps.

It is important to notice that the choice of the number of filters and their time scale is also arbitrary. So, for example, one can decide that the neuron may fire because of events that happened 10 seconds ago. Also, in implementing the GLM it is possible to use other shapes of filters.

[1](a.k.a. 'conditional intensity function' or 'firing probability')

[2]Pillow et al. Nature 2008

Implementation:

1. Create a Matlab file called GLMdata.mat that contains 2 sparse matrices:
    (a) **Raster**: a sparse matrix with one column for each neuron and one row for each time bin. When neuron 'a' gives a spike at time bin 'b' set Raster(b,a)=1;
    (b) **Stim**: a sparse matrix with one column for each stimulus type and one row for each time bin. If the value of stimulus 'a' is 's' at time bin 'b' set Stim(b,a)=s;
    (c) GLMdata.mat can also include a vector called '**BinsToIgnore**' that contains the time bins to ignore in the optimization. This is very useful for ignoring some segments in the data. For example, if one wants to focus on trials alone and ignore the ITIs.
2. Put the file in the same folder with the matlab functions:
    (a) FitGLM.m
    (b) minusLogL.m
    (c) makeBasis_PostSpike.m
    (d) sameconv.m
3. Set the value of the time bin duration '**dt**'.
4. Set the value of '**UnitNum**' that indicates which unit (column in **Raster**) to model.
5. Decide which kind of filter bases you need.
    (a) How many 'bumps' in each basis ?
    (b) How long after the event should the first bump occur?
    (c) How long after the event should the last bump occur?
    (d) (For the neuron's influence on itself) how long is the refractory period?
    (e) How evenly should the bumps be spaced?
6. Enter these parameters in the file FitGLM.m for the filter bases that couple the neuron to itself, to other neurons and to external stimuli.
7. Run FitGLM.m

Outputs:

The script 'FitGLM.m' creates several outputs and also plots them as figures:

1. OwnFilter - A vector of the neuron's self influence filter.
2. OtherNeuronsFilters - A matrix with the filters of other neurons' influence as its columns.
3. StimuliFilters - A matrix with the filters of the stimuli influence as its columns.
4. BaseLineRate - The neuron's baseline firing rate in the abscence of other neurons' spikes or external stimuli.
5. Lambda - The optimized time varying instantaneous rate. (The expected spikes per bin at time t is $\lambda(t) \cdot dt$)

Math appendix:

Definitions:

- Time is binned using segments of length Δ. To make things simple we write t instead of counting $n \cdot \Delta$ using the integer n.
- $x\left(t\right)$ - is the time series of the neuron's spikes.
- $h_x\left(\tau\right)$ - is the self influence filter.
- $\left\{y_i\left(t\right)\right\}_{i=1}^{N_o}$ - are the time serieses of the other, N_o, neurons in the recording session.
- $h_y^i\left(\tau\right)$ - are the other neurons' influence filter.
- $\left\{s_j\left(t\right)\right\}_{j=1}^{N_s}$ - are the time serieses of the N_s stimuli in the recording session.
- $h_s^j\left(\tau\right)$ - are the stimuli influence filter.
- γ - is the baseline firing coefficient.

The implemented code assumes the following functional dependence:

$$\lambda\left(t\right) = \exp\left\{\gamma + x * h_x + \sum_i^{N_o} y_i * h_y^i + \sum_j^{N_s} s_j * h_s^j\right\}$$

Using other non-linearities instead of 'exp' require a small change in the code and will increase optimization time a little.

The optimization is made by taking the spiking probability, $\lambda\left(t\right) \cdot \Delta$, as independent Bernoulli trials (it is more accurate to use the Poisson distribution and would not slow the calculation much). The log-likelihood function is:

$$\log L = \sum_{\{t_{sp}\}} \lambda\left(t_{sp}\right) - \Delta \cdot \int \lambda\left(t\right) dt + C$$

where $\{t_{sp}\}$ are the set of times in which the neuron spiked and C represent all the constants that add to the function but do not influence the optimization (omitted in the code).

Regularizations:

The above form of the log-likelihood is not bounded or regularized. This can lead to undesired results (e.g. getting $\lambda \cdot \Delta > 1$ for some time bins). Use the Poisson distribution for more accurate results.

Regulariztion deals with the number of parameters and their amplitude. The users are encouraged to insert their own regularization forms. (See Pillow et. al)