# General Form

Since we are now interested in dispersion, we need our time bin be large enough to allow multiple spikes. And we also need independent observations. Denote the number of spikes at step in neuron be and all observations at step as

If we only allow be 1 or 0, then the distribution is forced to be Bernoulli or approximately Poisson because of small intervals.

Denote , . Further denote , and  to help with derivation. Defining is to avoid tensor product (or write it out explicitly).

Then the likelihood is:

, where

And the log-likelihood is:

Then follow the same rationale as (A.3) in Eden’s paper:

Take Log on both sides:

Differentiate once and twice with respect to :

Evaluating these 2 derivatives at yields:

Since

only four pieces remain:

Now, we further need to write , , , and explicitly.

OK, put them altogether…

To be more explicit, we can write things out

So,…

When and , then this is exactly as derived in Eden’s paper.

# Fisher Scoring

Since the observed information include , it is not robust to outliers. Replacing observed information by fisher information will stabilize the algorithm, although we will sacrifice the efficiency a bit.

Pretty clean.

# In Context of Linear Regression

Let

Then , , and

For the fisher scoring version:

# Generalized Count Distribution

Different from Gao et el., I don’t include intercept into . Actually, I firstly consider things more general than linear regression form, i.e. is not necessarily .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Support | pdf |  |  |
| Binomial |  |  |  |  |
| Negative Binomial |  |  |  |  |
| Poisson |  |  |  |  |
| COM-Poisson |  |  |  |  |

Further, I write , where is the general dispersion parameter.

Denote and define dispersion parameter as . Therefore and denote . Further, denote . is a function that only depends on .

So the general filtering:

As previous

For , generally for nearly any real-valued () function . Denote , then and

Then,

Since and ,

Therefore,

For , . Generally, for nearly any real-valued () functions and .

Therefore,

(1) :

(2) and :

(3 ) :

By

We can get:

Write them together:

The fisher scoring will save our life (A LOT!)

Special cases:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Binomial |  |  | 0 | 0 | 0 |
| Negative Binomial |  |  |  |  |  |
| Poisson |  |  | 0 | 0 | 0 |
| COM-Poisson |  |  |  |  |  |

The COM-Poisson example is seen in previous derivation.

(For negative binomial, can you find a better , such that is not included in ? But if we just use Fisher scoring, that would not be a big issue, since)

The moments are evaluated numerically, if there’s no closed form.

No need for Fisher scoring:

# Specify a form of for approximation?

For example, let and , then . Well, I don’t think it will be very useful… Plot against (it seems COM-Poisson already did a good job)

