**In short, I think the problem is not the single observation but is the outlier.**

First, the convexity is not a problem for me. The natural parameters for COM-Poisson is . Since,

, it’s log-concave in terms of natural parameters. Loosely speaking, since for keeps the sign for hessian, so the convexity is not broken. We can get the hints from the CMP regression: if the log-convexity is not hold for and , then all previous researches about CMP regression are kind of problematic (they just do different kinds of MLE)… This gives a more detailed discussion <https://arxiv.org/pdf/1610.08244.pdf>.

I think the problem you see in the MLE simulation for single observation comes from . To give an intuition, let’s consider a rough approximation:

If we replace them with sample variance and expectation (assume MLE is close to MME), since and   is finite, we need shoot to infinity (under this bad approximation). But , that force to be super large. This is what you see in the simulation example, the MLE is super large in this case…

**However, in the adaptive filtering we have the prior prediction, which is equivalent to adding prior samples. So, we are actually dealing with prior samples + single observation when doing MLE.**

**OK, let me try to explain the “jump” in the filtering now.** When we take a look at the adaptive filtering update (i.e. n=1 in the addtfill\_compois\_v2.docx)

If the observation is much smaller than prior expectation, i.e. , then will be very small (roughly) or even to a negative value. This will cause 2 issues: 1) when but very small, will be large, i.e. we assign too much weight on observation. This leads to the “jump”; 2) when , the covariance matrix is even negative-definite! That’s invalid no matter there’s a “jump” or not. On the contrary, when , the is very small, so the outlier observation will not contribute a lot.

To kill the annoying “outlier”, one natural way is to think about the Fisher scoring, i.e. replace the observed information by expected (Fisher) information:

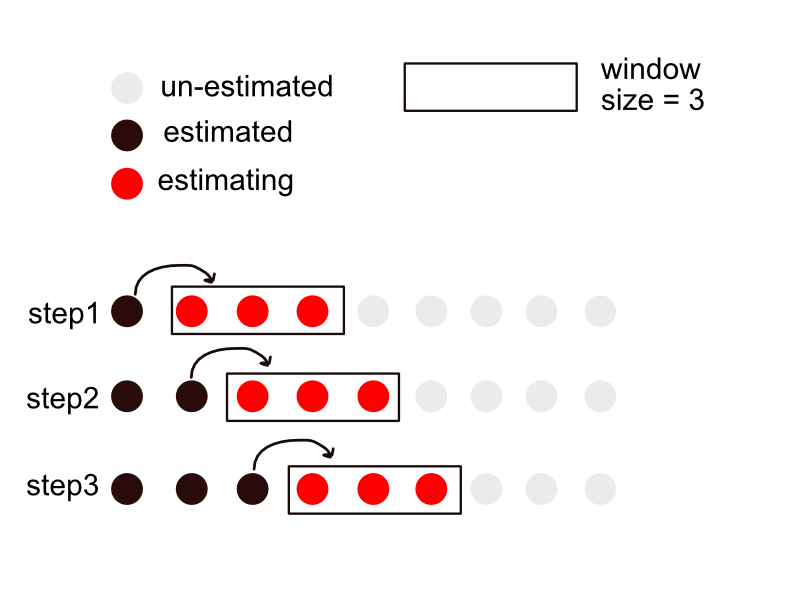
OK, now we will always have a positive-definite posterior variance, and we never need to worry about the “outliers”, by sacrificing some information in the observation. There’s another potential problem, if we initialize badly, the Fisher scoring version may not “adjust” itself quickly, and this might lead to singular .

To compare the raw adaptive filtering and the Fisher scoring, I use the **two case 3 as the examples**. Here, T = 10 and Q is set as [1e-3, 1e-3]. These two are initialized by the corresponding smoothing version.

|  |  |
| --- | --- |
| “raw” adaptive filtering | Fisher scoring |
|  |  |
|  |  |

OK, “Fisher scoring” resolves the “jump”/ robustness problem well, but it is not sensitive enough for observation: it just responds too slow.

Here, I tried to balance the robustness and sensitivity by using future observations. The basic idea is illustrated in the following figure:



Each dot represents the we want to estimate. Basically, in each iteration, I use observation in the window to facilitate the estimation by assuming state vectors in the window are the same. After estimation, I only let the first position in the window keep the estimates.

OK, let’s see if it works. Still use case 3 as the example, set the window size k = 1 (original), 5, 10 and 100 (crazy).

|  |  |  |
| --- | --- | --- |
|  | “raw” adaptive filtering | Fisher scoring |
| K=1 |  |  |
| K=5 |  |  |
| K=10 |  |  |
| K=100 |  |  |

OK, it works, although not elegant… We can see that: 1) the algorithm is not very sensitive to the window size, but when the size is too large the assumption “same state vectors in the window” no longer holds; 2) “Fisher scoring” doesn’t sacrifice a lot of information when we use the window. (This is as expected, since as by WLLN).

**Maybe, to keep robustness while not loose too much sensitivity, we can use the window + fisher scoring?**

When using the window, the window size becomes another tuning parameter. However, since the algorithm is not sensitive to it, we can just search it in a rough grid. Still use case 3 as the example, here I just use window + fisher scoring.

I first search across k = [1, linspace(5, 30, 6)] by maximizing the prediction likelihood, with Q = [1e-4, 1e-4]. (Maybe this time we should use filtering likelihood? Let me think it further). Plot the prediction log-likelihood against the window size:



Then turn on the Q-tuning under k=10, and do both filtering and smoothing:

