**Update: a new way to handle missing, as shown in** [**NA2**](https://github.com/weigcdsb/COM_POISSON/tree/main/demo/v1/NA2)**. Now the NAs will not contribute to gradients and hessian. As a result, I don’t need to do interpolation for held-out anymore. Everything is done when fitting the model.**

**The followings are old results.**

After thinking twice, I’m not very comfortable about sampling with equal space. So, I modified all code to handle missing. The lag-h state space equation is (i.e. h-1 NAs):

To help checking, just show the derivation…

# Neuron 13

Code: [v1\_comparison\_v4.m](https://github.com/weigcdsb/COM_POISSON/blob/main/demo/v1/v1_comparison_v4.m)

Use null Poisson as baseline, the bits/spike for different models:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dCMP-(5,3) | dCMP-(5,1) | dCMP-(5), constant nu | dPoi-(5) | sCMP-(5,3) | sCMP-(5,1) | sPoi-(5) |
| 0.107 | 0.1059 | 0.1043 | 0.078 | 0.0838 | 0.0831 | 0.0489 |
| 0.1032 | 0.1031 | 0.1009 | 0.0688 | 0.0902 | 0.0897 | 0.0523 |

The bits/trial for different models:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dCMP-(5,3) | dCMP-(5,1) | dCMP-(5), constant nu | dPoi-(5) | sCMP-(5,3) | sCMP-(5,1) | sPoi-(5) |
| 10.2082 | 10.1084 | 9.9512 | 7.4377 | 7.994 | 7.9288 | 4.668 |
| 10.0056 | 10.0022 | 9.7897 | 6.676 | 8.7463 | 8.7004 | 5.0676 |

# All 74 neurons

Code: [comparison\_all2.m](https://github.com/weigcdsb/COM_POISSON/blob/main/demo/v1/comparison_all2.m)

Now I monitor the NR, when it hit the max iteration (1000), it will output an error. This ensures the convergence. Neuron 9, 31, 37, 61, 64, 68 failed, I temporarily exclude them and will check what’s happening later. The bits/spike for remaining 68 neurons (showing Q1, median and Q3):



A few looks strange (negative dPoisson). **However, the convergence is ensured in NR step**. I guess this might be caused by bad Q selection? I don’t know, let me check later…

After further excluding these four neurons (43, 50, 60, 67) with negative dPoisson bits/spike, the results for remaining 64 neurons:



The dCMP-(5)-constant-nu looks better than dCMP-(5,1) sometimes? The following table shows the **training** bits/spike results for all these 64 neurons (yellow = neuron 13, grey = dCMP-(5)-constant-nu better):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| dCMP-(5,3) | dCMP-(5,1) | dCMP-(5), constant nu | dPoi-(5) | sCMP-(5,3) | sCMP-(5,1) | sPoi-(5) |
| 0.6633 | 0.651 | 0.683 | 0.1597 | 0.6053 | 0.6053 | 0.0844 |
| 0.4277 | 0.4181 | 0.4203 | 0.1196 | 0.397 | 0.397 | 0.0543 |
| 0.6291 | 0.6266 | 0.6277 | 0.2687 | 0.6224 | 0.6224 | 0.2556 |
| 0.1587 | 0.1575 | 0.1637 | 0.1084 | 0.1103 | 0.1103 | 0.0465 |
| 0.4305 | 0.4213 | 0.4331 | 0.1339 | 0.414 | 0.414 | 0.1009 |
| 0.5845 | 0.5704 | 0.5922 | 0.1136 | 0.5589 | 0.5589 | 0.0716 |
| 0.8299 | 0.807 | 0.8307 | 0.0384 | 0.825 | 0.825 | 0.0294 |
| 0.5757 | 0.5609 | 0.5805 | 0.1836 | 0.485 | 0.485 | 0.0402 |
| 0.5018 | 0.4901 | 0.5059 | 0.0856 | 0.4577 | 0.4577 | 0.0125 |
| 0.967 | 0.9662 | 0.9698 | 0.5234 | 0.9638 | 0.9638 | 0.507 |
| 0.2941 | 0.2918 | 0.2955 | 0.0533 | 0.28 | 0.28 | 0.0309 |
| 0.1119 | 0.1114 | 0.1087 | 0.0779 | 0.0891 | 0.0886 | 0.05 |
| 0.1486 | 0.1483 | 0.1708 | 0.1012 | 0.1367 | 0.1365 | 0.0504 |
| 0.3438 | 0.331 | 0.3475 | 0.0761 | 0.3051 | 0.3051 | 0.02 |
| 0.5961 | 0.5813 | 0.5967 | 0.0877 | 0.5789 | 0.5789 | 0.0226 |
| 0.0682 | 0.0666 | 0.0682 | 0.0502 | 0.0585 | 0.0576 | 0.0329 |
| 0.3781 | 0.3696 | 0.3817 | 0.0776 | 0.3407 | 0.3407 | 0.0093 |
| 0.3398 | 0.3297 | 0.3432 | 0.0615 | 0.3268 | 0.3268 | 0.0082 |
| 0.3643 | 0.3526 | 0.3705 | 0.0334 | 0.3554 | 0.3554 | 0.0177 |
| 0.4408 | 0.4253 | 0.4514 | 0.1356 | 0.3769 | 0.3769 | 0.0309 |
| 0.3147 | 0.308 | 0.3198 | 0.1022 | 0.2872 | 0.2872 | 0.0228 |
| 0.2798 | 0.2718 | 0.2831 | 0.078 | 0.2543 | 0.2543 | 0.0121 |
| 0.2161 | 0.2119 | 0.2236 | 0.1204 | 0.1684 | 0.1681 | 0.0428 |
| 0.4124 | 0.4113 | 0.4157 | 0.1554 | 0.3228 | 0.3228 | 0.0435 |
| 0.2629 | 0.2578 | 0.28 | 0.1022 | 0.2246 | 0.2246 | 0.0446 |
| 0.5114 | 0.5106 | 0.5102 | 0.113 | 0.4453 | 0.4453 | 0.0282 |
| 0.4956 | 0.4859 | 0.4938 | 0.062 | 0.4704 | 0.4704 | 0.0155 |
| 0.6796 | 0.6716 | 0.6811 | 0.5226 | 0.6511 | 0.6479 | 0.4656 |
| 0.8169 | 0.8115 | 0.8187 | 0.4233 | 0.8159 | 0.8159 | 0.4224 |
| 0.6787 | 0.6557 | 0.6829 | 0.0688 | 0.6498 | 0.6498 | 0.0107 |
| 0.7783 | 0.7458 | 0.7859 | 0.1085 | 0.7261 | 0.7261 | 0.022 |
| 0.342 | 0.3332 | 0.3451 | 0.0956 | 0.2886 | 0.2886 | 0.0319 |
| 0.803 | 0.7768 | 0.8056 | 0.0528 | 0.7876 | 0.7876 | 0.0154 |
| 0.6649 | 0.6456 | 0.667 | 0.1383 | 0.6525 | 0.6525 | 0.119 |
| 0.3192 | 0.3185 | 0.3222 | 0.1516 | 0.2982 | 0.2982 | 0.1639 |
| 0.6003 | 0.5806 | 0.6035 | 0.0898 | 0.5787 | 0.5787 | 0.0254 |
| 0.6451 | 0.6452 | 0.6443 | 0.342 | 0.6177 | 0.6177 | 0.2991 |
| 0.5702 | 0.5506 | 0.5765 | 0.1541 | 0.5102 | 0.5102 | 0.0172 |
| 0.3831 | 0.3756 | 0.3867 | 0.0781 | 0.3383 | 0.3383 | 0.0166 |
| 0.1334 | 0.1363 | 0.1366 | 0.0709 | 0.123 | 0.1221 | 0.0518 |
| 0.1273 | 0.1266 | 0.2195 | 0.2104 | 0.0297 | 0.0292 | 0.0181 |
| 0.0884 | 0.0828 | 0.0831 | 0.0814 | 0.0765 | 0.0705 | 0.0683 |
| 0.2334 | 0.2495 | 0.2455 | 0.1912 | 0.169 | 0.1672 | 0.0824 |
| 0.7411 | 0.7218 | 0.7442 | 0.0812 | 0.7132 | 0.7132 | 0.0272 |
| 0.2885 | 0.2855 | 0.2902 | 0.1077 | 0.2271 | 0.2271 | 0.0102 |
| 0.7021 | 0.6777 | 0.6997 | 0.0706 | 0.6818 | 0.6818 | 0.0373 |
| 0.6171 | 0.5997 | 0.6194 | 0.054 | 0.5983 | 0.5983 | 0.0227 |
| 0.4975 | 0.484 | 0.5025 | 0.0684 | 0.4637 | 0.4637 | 0.0235 |
| 0.2658 | 0.2648 | 0.2678 | 0.0173 | 0.2603 | 0.2603 | 0.0157 |
| 0.8526 | 0.8338 | 0.8552 | 0.0481 | 0.8432 | 0.8432 | 0.0263 |
| 0.5706 | 0.5593 | 0.5776 | 0.0387 | 0.5487 | 0.5487 | 0.0075 |
| 0.541 | 0.5244 | 0.5434 | 0.1186 | 0.4948 | 0.4948 | 0.032 |
| 0.7075 | 0.7001 | 0.71 | 0.1005 | 0.6994 | 0.6994 | 0.0806 |
| 0.4349 | 0.4318 | 0.4383 | 0.0761 | 0.4382 | 0.4382 | 0.0829 |
| 0.623 | 0.6212 | 0.6288 | 0.187 | 0.5998 | 0.5998 | 0.1548 |
| 0.6203 | 0.6106 | 0.6237 | 0.1226 | 0.6029 | 0.6029 | 0.0937 |
| 0.6649 | 0.6553 | 0.6432 | 0.0346 | 0.6628 | 0.6628 | 0.0349 |
| 0.4655 | 0.4565 | 0.4681 | 0.2382 | 0.4157 | 0.4157 | 0.149 |
| 0.3581 | 0.3501 | 0.3634 | 0.0629 | 0.342 | 0.342 | 0.0458 |
| 0.621 | 0.6071 | 0.625 | 0.0752 | 0.5916 | 0.5916 | 0.0174 |
| 0.4294 | 0.4156 | 0.4335 | 0.0652 | 0.4086 | 0.4086 | 0.0281 |
| 0.1019 | 0.0986 | 0.0966 | 0.067 | 0.0836 | 0.0821 | 0.047 |
| 0.44 | 0.4288 | 0.4438 | 0.0973 | 0.3913 | 0.3913 | 0.0197 |
| 0.6022 | 0.592 | 0.6109 | 0.3438 | 0.3972 | 0.3972 | 0.0099 |

Hmmm, it seems dCMP-(5)-constant-nu are usually better than the other 2 dCMPs. Neuron 13 is just the special case. Strange… (Maybe this is also caused by simplified Q selection? Q\_lam = diag([Q0, Q1, Q1,….]) and Q\_nu = diag([Q2, Q3, Q3,…]).)

Anyway, the bits/trial for these 64 neurons:



New:

Just use a more robust initial

initIdx = max(10\*nObs, find(cumsum(spk\_vec) > 200, 1, 'first'));

writematrix(spk\_vec(1:initIdx)', [usr\_dir '\Documents\GitHub\COM\_POISSON\runRcode\y.csv'])

writematrix(Xb(1:initIdx, :),...

[usr\_dir '\Documents\GitHub\COM\_POISSON\runRcode\X.csv'])

writematrix(Gb\_full(1:initIdx, :),...

[usr\_dir '\Documents\GitHub\COM\_POISSON\runRcode\G.csv'])

But there are still 3 neurons fail: neuron = 31, 38, 64.



