Simple nonlinear imputation

December 2020

1 Monte Carlo simulation (before cluster)

We are studying a data generating process and estimator from Example 2,

$$Z_i \sim U[-2,2]^{k_z} \tag{1}$$
$$X_i = Z_i\delta_x + \epsilon_i^x \tag{2}$$
$$M_i = \mathbf{1}[|Z_i\delta_m + \epsilon_i^m| > 0.8] \tag{3}$$
$$\epsilon_i^{m,x} \sim N[0,1] \text{ i.i.d, conditional on } X_i, Z_i \tag{4}$$
$$Y_i = \mathbf{1}[\alpha X_i + Z_i\gamma \geq U_i] \tag{5}$$
$$U_i \sim N[0,1] \text{ i.i.d, conditional on } X_i, Z_i, \tag{6}$$

in addition, the normal disturbance terms $U_i, \epsilon_i^m, \epsilon_i^x$ are independent of each other as well. The coefficient vector of interest (α, γ) is estimated by NLLS, which means that

$$g_0 = E\left[\begin{array}{c} X_i[Y_i - \Phi(\alpha X_i + Z_i\gamma)] \\ Z_i[Y_i - \Phi(\alpha X_i + Z_i\gamma)] \end{array}\right]$$

as we mentioned in section **??**. We also estimate the conditional probability density function of X_i at 400 grid points on $]-2,2[$ for every Z_i vector we observe when $M_i = 1$ by

$$\hat{f}_{x|z}(x,z) = \frac{\sum_j \prod_{k=1}^{k_z} K[(z_j^k - z^k)/h_{2,n}] \cdot K[(x_j - x)/h_{1,n}]}{\sum \prod_{k=1}^{k_z} K[(z_j^k - z^k)/h_{2,n}]}, \tag{7}$$

n	Infeasible GMM	Complete case GMM	Imputation GMM
1,000			
Mean	1.013, 0.504, -2.023	1.028, 0.509, -2.046	1.037, 0.526, -2.084
St. dev.	0.111, 0.095, 0.145	0.163, 0.142, 0.212	0.466, 0.265, 0.697
MSE	0.0144	0.0316	0.0268
2,000			
Mean	1.005, 0.505, -2.011	1.011, 0.508, -2.023	1.012, 0.514, -2.036
St. dev.	0.077, 0.065, 0.099	0.110, 0.096, 0.145	0.111, 0.075, 0.121
MSE	0.0067	0.0144	0.0114
4,000			
Mean	1.003, 0.502, -2.005	1.006, 0.503, -2.010	1.006, 0.507 -2.019
St. dev.	0.054, 0.046, 0.069	0.076, 0.069, 0.099	0.076, 0.052, 0.083
MSE	0.0033	0.0068	0.0053
8,000			
Mean	1.001, 0.501, -2.003	1.004, 0.502, -2.007	1.004, 0.503, -2.010
St. dev.	0.038, 0.032, 0.049	0.055, 0.048, 0.070	0.055, 0.036, 0.058
MSE	0.0016	0.0034	0.0026

Table 1: Monte Carlo simulations for $k_v = 2$ for the NLLS estimator in the probit model with 5,000 repetitions. The true values are $[1, 0.5, -2]]$, where the first coefficient corresponds to the missing variable X_i. Besides the means and standard deviations of the estimates we included the Mean Squared Error (MSE) once averaged over the three coefficient estimates.

where K is the Gaussian kernel, and h_1, h_2 bandwidth decrease with a -0.33 rate. After this, we define

$$\hat{e}(z_i; a, c) = z_i \left(y_i - \sum_{g=1}^{400} \Phi\left(a x_g + z_i c\right) \hat{f}_{x|z}(x_g, z_i) \right), \tag{8}$$

where x_g represent the gth point on the 400-grid we chose for the support of X_i.

This estimator has the same properties as the Nadaraya-Watson estimator for $\hat{e}$, but it has the added benefit that we do not have to recalculate the non-parametric part for every function evaluation in the numerical optimization procedure. It also turns out to have nicer finite sample performance. This way the code is close to $O(n)$ computational complexity in the range of sample sizes we considered in this simulation (rather than $O(n^2)$). For the sake of simplicity, we only target the γ coefficients and use the additive results in section **??**, while omitting the first moment from g_0, corresponding to X_i.

Table 1 gives the result for $k_z = 2$ with $\alpha = 1, \gamma = (0.5, -2)$ and 5,000 iterations. We can see how the imputation estimator has an overall better performance for the γ coefficients compared to the complete case estimator. The bias is slightly higher for the imputation estimator, due to the first stage, especially for the lower sample sizes, while the variance is much lower. The two estimator gives virtually the same results for the α coefficient. The estimators are $\sqrt{n}$-consistent as prescribed above $n = 2,000$. As for numerical stability, the BFGS algorithm with imputation estimator had trouble to find the minimum for $n = 1,000$ in one occasion, and resulted in an implausibly high coefficient estimate. We included this estimate value in the mean and standard deviation calculations, but excluded the iteration when evaluating the MSE.

NOTES:

- I found that directly estimating the conditional expectation of the moments is less computationally advantageous than doing the numerical integration above (which is somewhat unexpected). In addition, the second, seemingly more complicated option also has a better finite sample performance, it seems. The cost: need to choose 2 bandwidths instead of 1. Should I rewrite the estimation section to reflect the estimator here, or is it clear that this is really just a very similar approach?
- Is it ok to leave out the first moment from g_0 here? It is more compatible with Abrevaya and Donald + simpler. (Although technically speaking, I think you can make other simple arguments to estimate $E[Y_i X_i]$ separately (just conditioning on Y_i).
- Numerical instability: I haven't programmed the evaluation of the Jacobian and the Hessian into the code. My conjecture is that if this issue is due to numerical problems, this would help. The real question is: Is this a problem? Should we improve the code in this direction? Even if the answer is 'no', we may want to talk about how to communicate this issue.
- The simulation above ran for 36 hours with 10 i5 cores and 16 GB memory. This can get somewhat lower if the Jacobian + Hessian are evaluated. I expect that a single node on the cluster at least thirds computation time.
- As it is written, the code is efficient for these lower sample sizes. The first stage is using a nice little function to do matrix manipulation, but this function is going to switch from $O(n)$ to $O(n^2)$ complexity (and needs

much more memory, for some reason) above appx. 10,000 observations. This means that scaling to n=16,000 would have be more costly.

- Currently running "what happens" when $k_z = 5$ and the bandwidth is decreasing with a -0.33 rate, regardless. More problem with numerical stability at 1,000 already seen,[1] once the results are "cleaned" from suspiciously high/low values, the MSE for the imputation estimator is still slightly below the complete case estimator (but really only slightly). I would have expected slightly above, but this may be simulation error? (Also, 1,000 seemed to be still before the point asymptotics would start to kick in.) Including 3 more Z_i-s increased computation time, and the run will take around 48 hours, even after excluding the $n = 8,000$ sample size (which can only run with 4 cores and 16 GBs on my computer and the 5,000 iterations would take around 100 hrs - this probably can be sped up 7fold for the cluster though). All in all, a wishlist should be compiled for simulations. What do you wish for? :)

(this was Dec 2020)

2 UNR simulations

We also have the UNR simulation in January 2021, except we did 5,000 iterations, sample size 1-2-4,000; included Oracle results for 2,3,4 and 5 "Z"-s (one of them is the constant!).

We should see that for the Z=5 case things are not better anymore. Probably need to explain we still have root-n consistency for the imputation estimator, because of the weighting matrix!

The raw results are verbatim below the MSE table.

```
2 Z-s

beta= [0.5, -2]
alpha= 1
seed= 2433523
reps= 5000
direct:False
```

[1] The complete case estimator has some problems like this. The number of excluded iterations is around 200 this time, though.

```
n= 1000

means:
[[ 1.01439066  0.5077113  -2.02626132]
[ 1.02463808  0.51599755 -2.04811929]
[ 1.02721999  0.52126924 -2.06730093]
[ 1.02744845  0.52561572 -2.07421767]]

standard deviations:
[[0.1101572  0.09265875 0.14457649]
[0.16129516 0.14012859 0.2122271 ]
[0.16277967 0.10405336 0.18016047]
[0.16344342 0.10946485 0.18463714]]

This took 46635.64716243744 s



n= 2000

means:
[[ 1.00756409  0.50219724 -2.01415   ]
[ 1.01365273  0.50648401 -2.02758375]
[ 1.01473471  0.5092289  -2.03555785]
[ 1.01465123  0.51182362 -2.03969864]]

standard deviations:
[[0.0778399  0.065879   0.09879469]
[0.11021503 0.09688475 0.14479596]
[0.1105747  0.07143688 0.11924836]
[0.11072426 0.07481219 0.12228035]]

This took 90428.59637737274 s
```

```
n= 4000

means:
[[ 1.00241861  0.50250037 -2.00640034]
[ 1.00547112  0.50464767 -2.01249492]
[ 1.00604295  0.50600207 -2.01695352]
[ 1.00602551  0.50723998 -2.01894135]]

standard deviations:
[[0.05467132 0.04583565 0.06998306]
[0.07730954 0.0673744  0.1002694 ]
[0.07719927 0.04904421 0.08214253]
[0.07731308 0.05164758 0.08406368]]

This took 188247.8280761242 s


3 Z-s

beta= [0.5, -2, 1.2]
alpha= 1
seed= 2433523
reps= 5000
direct:False



n= 1000

means:
[[ 1.01733909  0.50748799 -2.02922031  1.21735961]
[ 1.03964554  0.52043198 -2.0720429   1.24331338]
[ 1.04659922  0.52429485 -2.09488189  1.25702206]
[ 1.04513322  0.54093334 -2.11340136  1.28377077]]
```

```
standard deviations:
[[0.11726256 0.09604072 0.15911832 0.12154718]
[0.1797207  0.15702465 0.24627595 0.19793294]
[0.18282035 0.10950621 0.20788941 0.14233428]
[0.18415238 0.12088293 0.21759271 0.15707991]]

This took 76655.7065961361 s



n= 2000

means:
[[ 1.00757038  0.50383891 -2.01566932  1.21016546]
[ 1.01494125  0.50865516 -2.03367499  1.22117445]
[ 1.01785413  0.51138741 -2.04458921  1.22880948]
[ 1.01654836  0.5224095  -2.05655383  1.24630799]]

standard deviations:
[[0.08091335 0.06631202 0.10877744 0.08390729]
[0.12188455 0.10583387 0.16211026 0.13279009]
[0.12247957 0.07262199 0.13235086 0.09305663]
[0.12291674 0.08009692 0.13700184 0.10163797]]

This took 127401.01496267319 s



n= 4000

means:
[[ 1.00424224  0.50274891 -2.00861362  1.20498507]
[ 1.00996358  0.50465937 -2.01833821  1.21136837]
[ 1.01145439  0.50651439 -2.02399403  1.21449611]
[ 1.01066273  0.51313837 -2.03158678  1.22541855]]

standard deviations:
```

```
[[0.05626844 0.04575098 0.07557173 0.05803291]
[0.08529223 0.07529992 0.11232224 0.09373892]
[0.08546765 0.05020719 0.09079608 0.06359679]
[0.08563915 0.05514322 0.09501318 0.07056298]]

This took 249020.63539409637 s


4 Z-s

beta= [0.5, -2, 1.2, -0.7]
alpha= 1
seed= 2433523
reps= 5000
direct:False



n= 1000

means:
[[ 1.02124124  0.50776913 -2.0391076   1.22429527 -0.7147819 ]
[ 1.0439207   0.51974554 -2.08655829  1.25238038 -0.73065132]
[ 1.05571459  0.52976592 -2.12110726  1.27527435 -0.7426668 ]
[ 1.10509051  0.57596206 -2.22508605  1.356705   -0.76675399]]

standard deviations:
[[0.11967458 0.09821092 0.15961672 0.12181884 0.08368603]
[0.18708826 0.16495896 0.25691246 0.20467911 0.13248178]
[0.19296063 0.1134623  0.21811564 0.14760399 0.10956413]
[2.86069222 1.22053192 4.0719307  2.24717299 1.39814556]]

This took 88795.59258174896 s



n= 2000
```

```
means:
[[ 1.00784571  0.50559481 -2.01853351  1.21099274 -0.7063667 ]
[ 1.01925346  0.51152461 -2.04111182  1.22445633 -0.71409324]
[ 1.02399421  0.51534631 -2.05624848  1.23429678 -0.71932715]
[ 1.0200414   0.53007038 -2.07017955  1.25702407 -0.71677156]]

standard deviations:
[[0.08483039 0.06748487 0.11044018 0.08518402 0.05918536]
[0.12435852 0.11144179 0.16789377 0.14076329 0.08973181]
[0.12553418 0.07431966 0.13562667 0.0959909  0.07185149]
[0.12550936 0.08368826 0.142854   0.10718826 0.07648801]]

This took 145436.29056882858 s



n= 4000

means:
[[ 1.00489871  0.50216096 -2.00949988  1.20621453 -0.70415826]
[ 1.01076652  0.50568142 -2.02152751  1.21266046 -0.70810677]
[ 1.01287459  0.50735177 -2.02833976  1.21789011 -0.71070734]
[ 1.01086453  0.51674262 -2.03781206  1.23256251 -0.70947631]]

standard deviations:
[[0.05855353 0.04673283 0.07686032 0.05917442 0.04028376]
[0.08678752 0.07564617 0.11888006 0.09609974 0.0620159 ]
[0.08738292 0.05061292 0.0924029  0.06497871 0.04852176]
[0.08744949 0.05683766 0.09798208 0.07288176 0.05155533]]

This took 282380.30895876884 s

5 Z-s

beta= [0.5, -2, 1.2, -0.7, 0.3, 1]
alpha= 1
```

```
seed= 2433523
reps= 5000
direct:False



n= 1000

means:
[[ 1.02725622  0.5176001  -2.05864265  1.23721569 -0.72106905  0.31020889
1.03182115]
[ 1.07123975  0.54008013 -2.14637137  1.29162637 -0.75003755  0.32440195
1.07719832]
[ 1.1921295   0.57989283 -2.33978741  1.39230093 -0.82495065  0.35665218
1.15228406]
[ 1.4996324   0.89112044 -3.08191948  1.97792953 -1.03554466  0.39465755
1.64884431]]

standard deviations:
[[ 0.12954025  0.10149575  0.17274183  0.12996276  0.09063694  0.07967804
0.12708357]
[ 0.21711901  0.18086076  0.30319765  0.23741623  0.14984034  0.13398944
0.2221814 ]
[ 2.97397908  0.87117646  4.53042755  2.25323743  1.68964042  0.78757515
1.65738204]
[ 7.01029255  3.51006939 12.13844962  7.76745485  4.69750453  1.91450644
6.03361828]]

This took 143902.393999815 s



n= 2000

means:
[[ 1.01470954  0.50724365 -2.0306106   1.21785328 -0.71151732  0.30471889
1.01636156]
```

```
[ 1.03106299  0.51723994 -2.06647681  1.24041647 -0.72394697  0.30909333
1.03496492]
[ 1.04045138  0.52138167 -2.08772044  1.25236737 -0.73119327  0.3131551
1.04509837]
[ 1.0138901   0.61210066 -2.17333147  1.3788245  -0.72449483  0.28192102
1.16517125]]

standard deviations:
[[0.08961579 0.07054348 0.11596102 0.08703768 0.06134015 0.05491414
0.0856286 ]
[0.13878747 0.1185509  0.18735632 0.15032367 0.09772463 0.08766628
0.14509982]
[0.14242275 0.07867273 0.15073804 0.10206377 0.07777906 0.06720136
0.09733127]
[0.14175696 0.08854494 0.16819501 0.11907604 0.08365428 0.07090414
0.10943478]]

This took 210180.16684031487 s



n= 4000

means:
[[ 1.00581683  0.50204505 -2.01101345  1.20655051 -0.70449117  0.30238619
1.00621672]
[ 1.0131092   0.50663293 -2.02959267  1.21728989 -0.71112976  0.30383917
1.01722718]
[ 1.01730848  0.50879099 -2.03796274  1.222628   -0.71391768  0.3060213
1.02041763]
[ 1.00034329  0.5753309  -2.10107245  1.31627904 -0.70864686  0.28371132
1.1078926 ]]

standard deviations:
[[0.06151435 0.04881332 0.08100703 0.06169968 0.04331079 0.03766353
0.06134309]
[0.09619565 0.08319213 0.12715765 0.10371681 0.06733829 0.05943197
```

```
0.09891667]
[0.09718725 0.05382577 0.09993611 0.068655   0.05333421 0.04481709
0.06706657]
[0.09628772 0.06013854 0.10843431 0.07732557 0.05616947 0.04730025
0.07356644]]

This took 370531.224098444 s
```