Some notes about the Fortran code in msda function

Jing Zeng

September 13, 2018

1 Summary

(1) The meaning of some parameters:

- `obj`: Value of the objective function

$$\sum_{k=2}^{K}\{\beta_k^T\Sigma\beta_k - \beta_k^T(\mu_k-\mu_1)\} + \lambda\sum_{k=2}^{K}||\boldsymbol{\beta_{\cdot k}}||_2$$

  . The difference ratio $(obj_{new} - obj_{old})/obj_{new} < sml$ is a criterion of converage

- `nk`: K-1
- `sigma`: $\hat{\Sigma}$
- `delta`: $(\hat{\mu}_2 - \hat{\mu}_1, \cdots, \hat{\mu}_K - \hat{\mu}_1)$
- `pf`: a penalty vector used in code, is $\underbrace{(1, 1, \cdots, 1)}_{p}$ by default
- `pmax`: During the process of iteration with each λ, we have to make sure the number of non-zero group variables $ni < pmax$, otherwise the process will stopped and return error code $-10000 + l$ where l is the order of current λ
- `dfmax`: With each λ, if the converage is achieved and $ni < pmax$ thorough the whole process, we still need to check the number of non-zero variables, i.e. me, since some non-zero variables may degenerate to 0 again while ni kept unchanged. If $me > dfmax$, which means the number of non-zero variables is too large, error code $-20000 + l$ is returned. And by default $dfmax = nobs$, which indicates that typically the number of non-zero variables need to be less than the number of observations.
- `nlam`: the number of λ candidates.
- `flmin`: used to give a factor which help generate the sequence of λs
- `ulam`: just the sequence of λs. It will be used only if a sequence of self-defined λs is given
- `eps`, `sml`: two converage thresholds
- `maxit`: the maximal iterations
- `verbose`: should we print the process or not

- `nalam`: how many λ candidates are used. Since with some λ, there will pop out an error code, so the subsequent λs won't be used
- `theta`: $(\Sigma^{-1}(\mu_2-\mu_1),\cdots,\Sigma^{-1}(\mu_K-\mu_1))^T \in \mathbb{R}^{(K-1)\times p}$. The order of columns are shuffled, we need to use function `formatoutput` to reorder it.
- `ni`: The number of all the appeared non-zero variables during the process of converage.
- `me`: The number of all the non-zero variables in the end
- `m`: the positions of all the appeared non-zero variables, can be used to reorder the `theta`
- `ntheta`: a set of ni for each visited λs
- `alam`: a set of visited λs
- `npass`: iterations
- `jerr`: the error code.
  - 0: success
  - -10001: $ni > pmax$ during the process with the first λ candidate,
  - -20001: $me > dfmax$ with the first λ candidate
  - 10000: $maxval(pf) <= 0$
  - $-l$: $npass > maxit$ during the process with the lth λ candidate

(2) How to tune the parameter λ?

(a) $\lambda_1 = 9.9e30$ by default

(b) $alf = flmin**(1/(nlam-1))$ and calculate another quantity al based on `pf` and `delta`, then $\lambda_2 = al \times alf$

(c) $\lambda_{k+1} = \lambda_k \times alf, k = 2,\cdots,nlam-1$

(3) During the process of converage with each λ candidate, or say the outer loop, the only exit of this loop is $ni > pmax$ or converage achieved

(4) Outside the outer loop, if the new `theta` and the old `theta` are not very close, we can still use the criterion $(obj_{new} - obj_{old})/obj_{new} < sml$ to give it the second chance. If this condition holds, then we still think it as converged.

(5) Here are some outputs I got when error codes are thrown:

- `ni = 391, me = 320, pmax = 440, dfmax = 210, jerr = -20001`
- `ni = 441, pmax = 440, jerr = -10001`

2 How to compile a Fortran code

(1) In R:

(i) Run `module load intel` and `module load R` in bash

(ii) Run command `R CMD SHLIB fortran_file_name` in bash environment to compile Fortran file and it will generate a `.so` file and a `.o` file.

(iii) Add `dyn.load(so_file_name)` in Rscript file and with `.Fortran` function the Rscript file will apply the Fortran file.

(2) In bash:

(i) Run `module load intel` and `module load R` in bash

(ii) Write a Fortran file `.f`

(iii) Run command `gfortran fortran_file_name [-o output_file_name]` in bash environment to compile Fortran file and it will generate a `.out` file.

(iv) Run `./output_file_name` and the output will printed out.

Note:

(a) We can add `WRITE (*,*) VARIABLE_NAME` in Fortran file so the values of corresponding variable will be printed out.

(b) `R CMD BATCH` will generate another output file `.Rout`.