MultiSnpnet Vignette

Junyang Qian, Yosuke Tanigawa, Manuel A. Rivas, and Trevor Hastie

2020-06-08

Introduction

multiSnpnet is a package that is used to fit sparse reduced rank regression for multiple responses on big genomics data. It optimizes the following objective function:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{Y}) - \mathcal{P}_\Omega(\mathbf{X}\mathbf{U}\mathbf{V}^\top)\|_F^2 + \lambda\|\mathbf{U}\|_{1,2}, \\ \text{s.t.} \quad & \mathbf{V}^\top\mathbf{V} = \mathbf{I}, \end{aligned}$$

where $\mathcal{P}_\Omega$ is a projection operator for missing values in $\mathbf{Y}$, i.e. $\mathcal{P}_\Omega(\mathbf{Y})_{ij} = \mathbf{Y}_{ij}$ if $\mathbf{Y}_{ij}$ is not missing and 0 otherwise, and $\|\mathbf{U}\|_{1,2} = \sum_{j=1}^{p} \|\mathbf{U}_{j\cdot}\|_2$.

We assume the data are stored in .pgen/.pvar/.psam format by the PLINK library. The potential training/validation split can be specified by a separate column in the phenotype file.

The most essential parameters in the core function `multisnpnet` include:

- `genotype_file`: the PLINK 2.0 pgen file that contains genotype. We assume the existence of genotype_file.{pgen,pvar.zst,psam}.
- `phenotype_file`: the path of the file that contains the phenotype values and can be read as a table. Missing values are expected to be encoded as -9.
- `phenotype_names`: names of the phenotypes. Must be consistent with the column names in the phenotype file.
- `covariate_names`: a character vector containing the names of the covariates included in the fitting, whose coefficients will not be penalized. The names must exist in the column names of the phenotype file. Can be an empty vector.
- `rank`: target rank of the reduced rank model.
- `validation`: whether to evaluate the model performance on the validation set.
- `split_col`: the column name in the phenotype file that specifies the membership of individuals to the training or the validation set. The individuals marked as `"train"` and `"val"` will be treated as the training and validation set, respectively. If not specified, all the qualified samples will be treated as training samples.
- `mem`: Memory (MB) available for the program. It tells PLINK 2.0 the amount of memory it can harness for the computation. IMPORTANT if using a job scheduler.
- `batch_size`: the number of variants used in each batch screening and should be chosen based on the problem size and memory available. Default is 100.
- `save`: a boolean indicating whether to save intermediate results. If `TRUE`, `results.dir` in the `configs` list should be provided.
- `configs`: a list of additional configuration parameters. It can include:
  - `results.dir`: path to the directory holding intermediate results, if `save = TRUE`.
  - `nCores`: number of cores used for computation.

Some additional important parameters for model building include:

- `nlambda`: the number of lambda values on the solution path.

- `lambda.min.ratio`: the ratio of the minimum lambda considered versus the maximum lambda that makes all penalized coefficients zero.
- `standardize_response`: whether to standardize the responses before fitting to deal with potential different units of the responses. Default is `TRUE`.
- `prev_iter`: index of the iteration to start from (e.g. to resume a previously interrupted computation). Set `prev_iter = -1` if one wants to start from the last saved iteration.

There are other parameters that can be used to control the training behavior, such as `weight` to adjust the relative importance of the phenotypes, `p.factor` to control the priority of each variable entering the model.

A Simple Example

```
library(multiSnpnet)
```

```
configs <- list(
  plink2.path = "plink2",    # path to plink2 program
  zstdcat.path = "zstdcat"   # path to zstdcat program
)
# check if the provided paths are valid
for (name in names(configs)) {
  tryCatch(system(paste(configs[[name]], "-h"), ignore.stdout = T),
    condition = function(e) cat("Please add", configs[[name]], "to PATH, or modify the path in the confi
  )
}
```

```
genotype_file <- file.path(system.file("extdata", package = "multiSnpnet"), "sample")
phenotype_file <- system.file("extdata", "sample.phe", package = "multiSnpnet")

phenotype_names <- c("QPHE", "BPHE")
covariate_names <- c("age", "sex", paste0("PC", 1:10))

fit_multisnpnet <- multisnpnet(
  genotype_file = genotype_file,
  phenotype_file = phenotype_file,
  phenotype_names = phenotype_names,
  covariate_names = covariate_names,
  rank = 2,
  nlambda = 10,
  validation = TRUE,
  split_col = "split",
  batch_size = 100,
  standardize_response = TRUE,
  save = TRUE
) # we hide the intermediate messages
```

We can make prediction with the fitted object `fit_multisnpnet`. For example,

```
pred_multisnpnet <- predict_multisnpnet(
  fit = fit_multisnpnet,
  new_genotype_file = genotype_file,
  new_phenotype_file = phenotype_file,
  covariate_names = covariate_names,
  split_col = "split",
  split_name = c("train", "val"),  # can also include "test" if such samples are available in the pheno
```

```
  binary_phenotypes = c("BPHE"))
```

We can extract the R-squared (R^2) for both phenotypes.

```
pred_multisnpnet$R2
#> $train
#>               1         2         3         4         5         6         7
#> QPHE 0.21159360 0.2452703 0.3584840 0.5061911 0.6357102 0.7290243 0.8079288
#> BPHE 0.09673542 0.1208299 0.2129555 0.3890419 0.5531091 0.6839201 0.7719748
#>               8         9        10
#> QPHE 0.8585059 0.9049677 0.9362294
#> BPHE 0.8408161 0.8876381 0.9280919
#>
#> $val
#>               1          2         3          4           5          6
#> QPHE 0.19453209 0.23122819 0.2997470 0.30859783  0.26526204  0.1846543
#> BPHE 0.06171918 0.08469849 0.1076595 0.06737471 -0.02857306 -0.1504381
#>                7           8          9         10
#> QPHE  0.07731492 -0.003306899 -0.1423596 -0.2467398
#> BPHE -0.26877442 -0.422487367 -0.6299712 -0.8430128
```

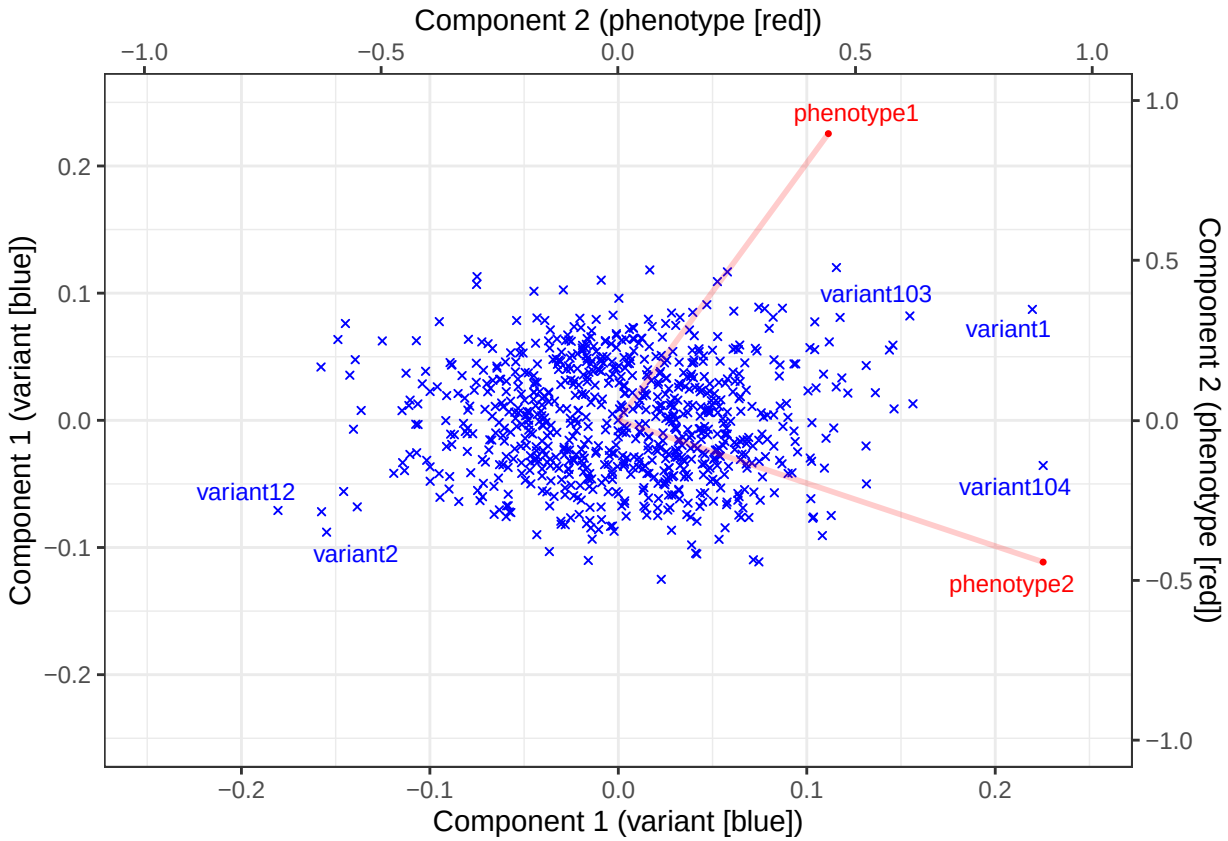
The negative validation R^2's are caused by overfitting. We can also obtain AUC for the specified binary phenotypes.

```
pred_multisnpnet$AUC
#> $train
#>               1         2         3        4         5         6        7
#> BPHE 0.6790872 0.7193103 0.8103339 0.907798 0.9599222 0.9873362 0.996743
#>               8 9 10
#> BPHE 0.9997484 1  1
#>
#> $val
#>               1         2         3         4         5         6         7
#> BPHE 0.6557377 0.6944927 0.6926796 0.6377326 0.6083705 0.5900632 0.5909949
#>               8         9        10
#> BPHE 0.5703457 0.5146937 0.5384402
```

Example of Biplot Visualization

The coefficent matrix C in the returned value from `multiSnpnet::multisnpnet()` has the dimension of $p \times q$. One can visualize this matrix with a biplot visualization.

```
fit_single <- fit_multisnpnet[[10]]
if (require(ggrepel))
plot_biplot(
  svd(t(as.matrix(fit_single$C))),
  label=list(
    'phenotype'=rownames(fit_single$A_init), # we should update this phenotype labels so that it matche
    'variants'=rownames(fit_single$C)
  )
)
#> Loading required package: ggrepel
#> Loading required package: ggplot2
```



By default, this function annotates 5 phenotypes and 5 variants based on the distance from the center of origin. One can annotate more points by specifying `n_labels` option.

Alternatively, you can generate an interactive plot with `plotly` with mouseover annotation of phenotypes and variants. In that case, you need to disable the static annotation by specifying `use_ggrepel=FALSE`.

```
# if (!require('plotly', character.only = TRUE)) {
#   install.packages('plotly', dependencies = TRUE)
#   library('plotly', character.only = TRUE)
# }
if (require(plotly)) {
  plot_biplot(
    svd(t(as.matrix(fit_single$C))),
    label=list(
      'phenotype'=rownames(fit_single$A_init), # we should update this phenotype labels so that it matc
      'variants'=rownames(fit_single$C)
    ),
    use_ggrepel=FALSE
  ) %>%
    plotly::ggplotly() %>%
    htmlwidgets::saveWidget('multisnpnet.biplot.html', selfcontained = F, libdir = "lib")
}
#> Loading required package: plotly
#>
#> Attaching package: 'plotly'
#> The following object is masked from 'package:ggplot2':
#>
#>     last_plot
```

```
#> The following object is masked from 'package:stats':
#>
#>     filter
#> The following object is masked from 'package:graphics':
#>
#>     layout
#> Warning: Ignoring unknown aesthetics: label

#> Warning: Ignoring unknown aesthetics: label
```

You can open the resulting html file (`multisnpnet.biplot.html`) with your favorite browser.