A introduction to `glmtlp`

Chunlin Li, Yu Yang, Chong Wu

October 14, 2021

Contents

Introduction

Glmtlp is a package that fits generalized linear models via penalized maximum likelihood. The regularization path is computed for the l0, l1, and TLP penalty at a grid of values (on the log scale) for the regularization parameter lambda or kappa (for l0 penalty). The algorithm is extremely fast. It fits linear and logistic regression models. The package includes methods for prediction and plotting, and functions for cross-validation.

The authors of glmtlp are Chunlin Li, Yu Yang, and Chong Wu, and the R package is maintained by Chunlin Li and Yu Yang. A Python version is under development.

This vignette describes basic usage of glmtlp in R.

Installation

Install the package from CRAN.

```
install.packages("glmtlp")
```

Quick Start

In this section, we will go over the main functions and outputs in the package.

First, we load the `glmtlp` package:

```
library(glmtlp)
```

```
## Loading required package: foreach
```

```
## Loading required package: parallel
```

We load a simulated dataset with continuous response to illustrate the usage of gaussian linear regression.

```
data(gau_data)
X <- gau_data$X
y <- gau_data$y
```

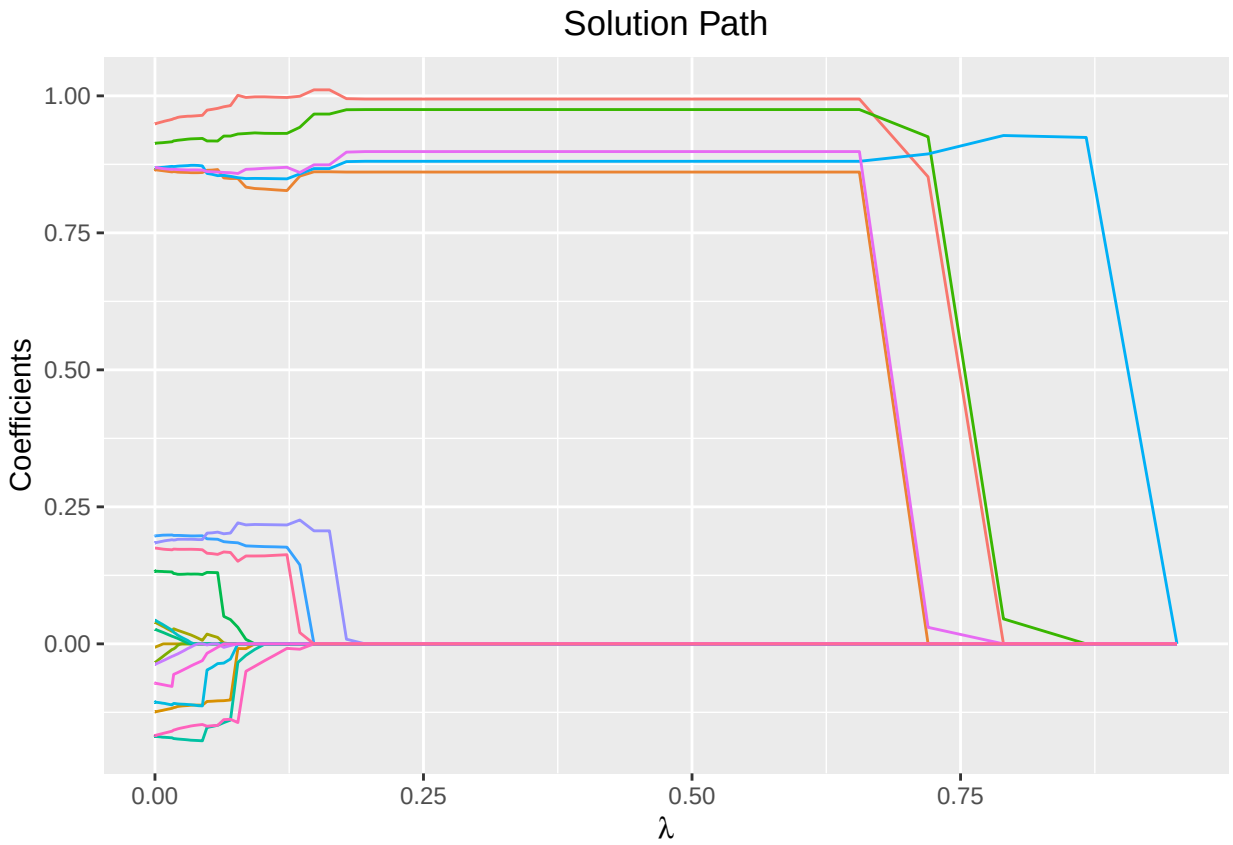
We fit three models by calling `glmtlp` with `X`, `y`, `family="gaussian"` and three different `penalty`.

```
fit <- glmtlp(X, y, family = "gaussian", penalty = "tlp")
fit2 <- glmtlp(X, y, family = "gaussian", penalty = "l0")
fit3 <- glmtlp(X, y, family = "gaussian", penalty = "l1")
```

`fit` is an object of class `glmtlp` that contains all the relevant information of the fitted model for further use. Users can apply `plot`, `coef` and `predict` to the fitted objects to get detailed results.

We can visualize the coefficients by executing the `plot` method:

```
plot(fit, xvar = "lambda")
```



The output is a `ggplot` object. Therefore, the users are allowed to make further modifications on the plot to suit their own needs. The plot shows the solution path of the model, with each curve corresponding to a variable. Users may also choose to annotate the curves by setting `label=TRUE`. Note that for "l1" or "tlp" penalty, `xvar` could be chosen from c("lambda", "log_lambda", "deviance", "l1_norm"), and for "l0" penalty, `xvar` could be chosen from c("kappa", "log_kappa").

We can use the `coef` function to obtain the fitted coefficients. By default, the results would be a matrix, with each column representing the coefficients for every λ or κ. The users may also choose to input the desired value of λ or κ

```
coef(fit)
```

```
...
##             0.951497   0.866968   0.789949    0.719772   0.655830    0.597568
## intercept -0.4014902 -0.2241503 -0.2295026 -0.19900287 -0.0146152 -0.01462573
## V1         0.0000000  0.0000000  0.0000000  0.85225605  0.9942131  0.99416056
## V2         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
## V3         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
## V4         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
```

```
## V5         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
## V6         0.0000000  0.0000000  0.0000000  0.03008658  0.8983170  0.89828772
## V7         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
## V8         0.0000000  0.0000000  0.0000000  0.00000000  0.0000000  0.00000000
...
```

```
coef(fit, lambda = 0.1)
```

```
...
##    intercept           V1           V2           V3           V4           V5
##  0.007500318  0.998208925  0.000000000  0.177556932  0.217623555  0.000000000
##           V6           V7           V8           V9          V10          V11
##  0.867573881  0.000000000 -0.033089363  0.160492880  0.830149182  0.000000000
##          V12          V13          V14          V15          V16          V17
##  0.000000000  0.000000000  0.000000000  0.931826941  0.000000000  0.000000000
##          V18          V19          V20
## -0.002744157  0.000000000  0.849237391
NA
NA
...
```

In terms of prediction, the users need to input a design matrix and the type, as well as the desired level of regularization parameters.

```
predict(fit, X[1:5, ], lambda = 0.1)
```

```
## [1] -1.5747735  0.3901481  0.1573404 -0.6070089  2.3966693
```

Cross-validation can be implemented by `cv.glmtlp` to find the best regularization parameter.

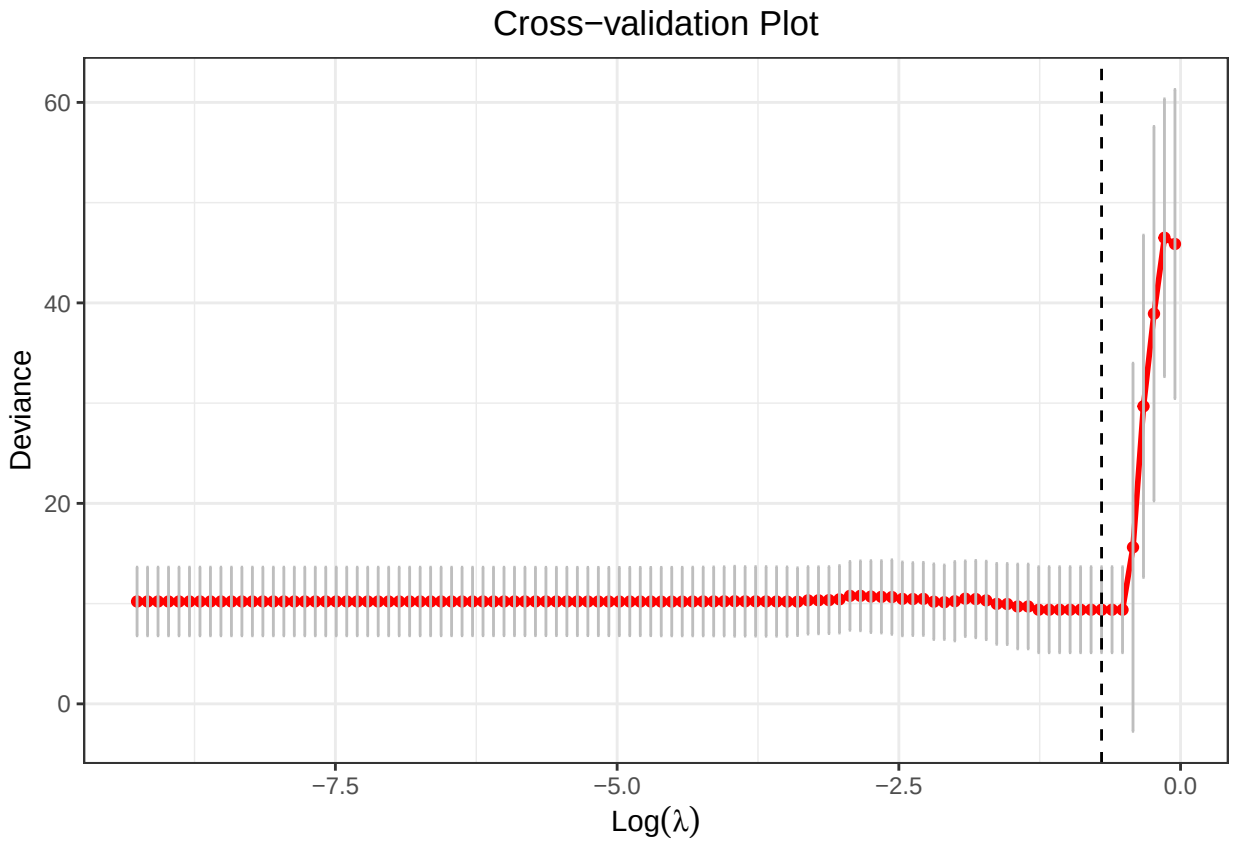
```
cv.fit <- cv.glmtlp(X, y, family = "gaussian", penalty = "tlp")
```

`cv.glmtlp` returns a `cv.glmtlp` object, a list with all the ingredients of the cross-validated fit. Users may use `coef`, `predict`, and `plot` to further check the cross-validation results.

```
coef(cv.fit)
```

```
##   intercept          V1          V2          V3          V4          V5
## -0.01462029  0.99418110  0.00000000  0.00000000  0.00000000  0.00000000
##          V6          V7          V8          V9         V10         V11
##  0.89829839  0.00000000  0.00000000  0.00000000  0.86092727  0.00000000
##         V12         V13         V14         V15         V16         V17
##  0.00000000  0.00000000  0.00000000  0.97498133  0.00000000  0.00000000
##         V18         V19         V20
##  0.00000000  0.00000000  0.88058038
```

```
plot(cv.fit)
```



This plot is a `ggplot` object and the users are allowed to make further modifications on it.

References

Shen, Xiaotong, Wei Pan, and Yunzhang Zhu. "Likelihood-based selection and sharp parameter estimation." *Journal of the American Statistical Association* 107.497 (2012): 223-232. https://doi.org/10.1080/01621459.2011.645783.

Tibshirani, Robert, et al. "Strong rules for discarding predictors in lasso-type problems." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 74.2 (2012): 245-266. https://doi.org/10.1111/j.1467-9868.2011.01004.x.

Yang, Yi, and Hui Zou. "A coordinate majorization descent algorithm for l1 penalized learning." *Journal of Statistical Computation and Simulation* 84.1 (2014): 84-95. https://doi.org/10.1080/00949655.2012.695374.