Vignette: Multiknockoffs

Chrysovalantis Karypidis

Introduction

The knockoff filter (Barber and Candès (2015); Candès et al. (2018)) is a modern and powerful algorithm to control the false discovery rate (FDR)

$$\mathbb{E}\left[\frac{|\hat{\mathcal{S}} \cap \mathcal{H}_0|}{|\hat{\mathcal{S}}| + q^{-1}}\right] \leq q,$$

for a variety of different model classes, including complex machine learning models such as neural networks, boosting, random forests or high-dimensional linear penalization methods. The procedure constructs fake features (a knockoff matrix $\tilde{\mathbf{X}}$), that mimic certain correlation properties of the original variables. Since knockoffs behave similar to the original features but are known to be artificial null variables, they serve as a negative control group.
Running the vanilla knockoff algorithm multiple times on the same data results in different knockoffs and probably overlapping but slightly different selection sets $\hat{\mathcal{S}}$. These differences will yield to fluctuating power and FDP values across different runs which make inference more irreproducible. The package `multiknockoffs` contains several extension of the initial knockoff filter, the so-called multiple knockoff filters. These run the knockoff filter multiple times, each time with a different generated probabilistic knockoff matrix $\tilde{\mathbf{X}}$. Then, the multiple selection sets of each knockoff run are aggregated in a special way, depending on the multiple knockoff filter. Multiple knockoff filters aim to increase the power while approximately retaining FDR control. Furthermore, they also reduce the variability of both quantities to ensure more reproducible results. The three implemented multiple knockoff filters are union knockoffs (uKO) by Xie and Lederer (2021), p-value knockoffs (pKO) by Nguyen et al. (2020) and ADAGES by Gui (2020).
This vignette aims to explain how to use the functions of `multiknockoffs` in `R`, and not to present the mathematical details of each method. Hence, the user should be familiar with the theoretical details of the three procedures before the application. More information about each multiple knockoff method can be found in the papers listed above or in my Master's thesis (Chapter 5) that is uploaded on the corresponding GitHub repository. The thesis also contains a simulation comparison of the three different multiple knockoff filters regarding their power and empirical FDR control.

Installation

The `multiknockoffs` package can be directly installed in `R` with the devtools package by typing the following commands:

```r
devtools::install_github("cKarypidis/multiknockoffs")
```

The implemented functions of `multiknockoffs` also rely on the original `knockoff` package by Patterson and Sesia (2020), which should be pre-installed. The user should also know the basic arguments of their functions since we use same notation at some point.

List of functions

The package contains the following functions:

`agg.ADAGES`	Aggregation step ADAGES
`agg.ADAGES.mod`	Aggregation step modified ADAGES
`agg.pKO`	Aggregation step pKO
`agg.union`	Union of selection sets
`multi.knockfilter`	Determination K score matrices and selection sets
`multi.knockoffs`	Construction multiple knockoff matrices
`quantile.aggregation`	Helper function: Quantile aggregation of the p-values
`run.ADAGES`	Whole ADAGES procedure with multiple knockoffs
`run.pKO`	Whole pKO procedure with multiple knockoffs
`run.uKO`	Whole uKO procedure with multiple knockoffs

Union knockoffs (uKO)

For all upcoming explanations, we will use the following generated data to illustrate the application of the functions. We generate data with $(n, p) = (400, 200)$ and randomly placed $|\mathcal{S}_0| = 30$ true signals, i.e. with non-zero coefficients, among those. The remaining 170 ones are null variables with zero coefficients. Furthermore, the predictors $\mathbf{X}$ have a Toeplitz structure as covariance matrix, and the response is drawn according to the linear model $y = \mathbf{X}\beta + \epsilon$.

```
n <- 400; p <- 200; s_0 <- 30
amplitude <- 1; mu <- rep(0,p); rho <- 0.25
Sigma <- toeplitz(rho^(0:(p-1)))

X <- MASS::mvrnorm(n, mu, Sigma)
nonzero <- sample(p, s_0)
beta <- amplitude * (1:p %in% nonzero)
y <- X %*% beta + rnorm(n)
```

The whole knockoff aggregation scheme uKO is implemented by `run.uKO`, which constructs K knockoff matrices, runs K knockoff filters at different nominal levels q_k and aggregates the resulting selection sets by taking their union.

```
run.UKO(X, y, knockoffs = create.second_order, statistic = stat.glmnet_coefdiff,
        qk = "decseq", q = 0.2, K = 5, q_seq = NULL, offset = 1 , sets = FALSE)
```

The arguments that can be supplied to the function are:

- `X` the $n \times p$ design matrix and `y` and the $n \times 1$ response vector.
- `knockoffs` is the function for the knockoff construction. It must take the $n \times p$ data matrix as input and it must return a $n \times p$ knockoff matrix. The user can either choose a knockoff sampler of the `knockoff` package or define it manually. Default: `create.second_order` (see below).
- `statistic` is a function that computes the scores W_j. It must take the data matrix, knockoff matrix and response vector as input and it outputs a vector of computed scores. The user can either choose one score statistic from the `knockoff` package or define it manually. Default: `stat.glmnet_coefdiff` (see below).
- `qk` the sequence of nominal levels. The user can choose between the options `"decseq"` (default) for $q_k = q/2^{k-1}$ or `"ave"` for $q_k = q/K$.
- `q` is the nominal level for the FDR control. Default: 0.2.
- `K` is the number of knockoff runs. Default: 5.
- `q_seq` to define an own sequence supplied by the user, which has to match in length with the number of knockoff runs `K`. If this argument is specified, `qk` and `q` are ignored.
- `offset` either 0 (knockoff) or 1 (knockoff+). Default: 1.

- `sets` logical argument if the K selection sets of each knockoff run before the aggregation should be returned. Default: `FALSE`.

The argument `create.second_order` refers to the approximate second-order knockoff construction, which is also the default option in the `knockoff` package. Moreover, the default `stat.glmnet_coefdiff` corresponds to the Lasso coefficient difference (LCD) statistic

$$W_j = |\hat{\beta}_j(\lambda_{\text{CV}})| - |\hat{\beta}_{j+p}(\lambda_{\text{CV}})|.$$

To compute the LCD statistic, the function runs a Lasso regression with data $(y, [\mathbf{X}\ \tilde{\mathbf{X}}])$, computes the coefficients at the cross-validated λ, and takes their absolute differences for a particular original variable $\mathbf{X}_j$ and its knockoff $\tilde{\mathbf{X}}_{j+p}$.
The function `run.uKO` outputs a list with the aggregated selection set $\hat{\mathcal{S}}_q^U$, the number of specified knockoff runs K, the theoretical FDR bound $\sum_{k=1}^{K} q_k$ of uKO and (if specified) the individual selection sets of each knockoff run $\hat{\mathcal{S}}_{q_1}, \ldots, \hat{\mathcal{S}}_{q_K}$.
If we run `run.uKO` with the previously introduced data and default settings, i.e. the decreasing sequence $q_k = q/2^{k-1}$ and $K = 5$, we obtain

```
res.UKO <- run.UKO(X, y, sets = TRUE)
res.UKO
$Shat
[1]   1   8   9  27  48  49  55  58  72  73  80  83  91  97 108
 115 119 121 126 129 139 142 144 146
[25] 147 151 155 158 163 165 167 171 174 178 181 183 186

$K
[1] 5

$FDRbound
[1] 0.3875

$sets
$sets$S1
[1]   1   8   9  27  48  49  55  58  72  73  80  83  91  97 108
 115 119 121 129 139 142 144 146 147
[25] 151 155 158 163 165 167 171 178 181 183 186

$sets$S2
[1]   8   9  27  48  49  55  58  72  73  91  97 108 115 119 121
126 129 139 142 144 146 147 151 155
[25] 158 163 167 171 174 181 186

$sets$S3
integer(0)

$sets$S4
integer(0)

$sets$S5
integer(0)
```

The selection sets $\{\hat{\mathcal{S}}_{q_3}, \hat{\mathcal{S}}_{q_4}, \hat{\mathcal{S}}_{q_5}\}$ are empty. Since we have used the decreasing sequence, the nominal levels of knockoff runs 3–5 are comparably small ($q_3 = 0.05,\ q_4 = 0.025,\ q_5 = 0.0125$). Those runs are very strict such that no variables are selected.

P-value knockoffs (pKO)

Next, we can run the multiple knockoff filter pKO by the command `run.pKO`

```
run.pKO(X, y, knockoffs = create.second_order, statistic = stat.glmnet_coefdiff,
        q = 0.2, B = B, gamma = 0.3, offset = 1, method = "BH", pvals = FALSE)
```

The user can adjust the following parameters:

- `X` the $n \times p$ design matrix and `y` and the $n \times 1$ response vector.
- `knockoffs` is the function for the knockoff construction. It must take the $n \times p$ data matrix as input and it must return a $n \times p$ knockoff matrix. The user can either choose a knockoff sampler of the `knockoff` package or define it manually. Default: `create.second_order`.
- `statistic` is a function that computes the scores W_j. It must take the data matrix, knockoff matrix and response vector as input and it outputs a vector of computed scores. The user can either choose one score statistic from the `knockoff` package or define it manually. Default: `stat.glmnet_coefdiff`.
- `q` defines the nominal level for the FDR control. Default: 0.2.
- `B` is the number of knockoff runs. Default: 25.
- `gamma` is a value between $(0, 1)$ which defines the quantile value used for the aggregation. If `gamma = NULL`, the adaptive search by Meinshausen et al. (2009) is used. Default: 0.3.
- `offset` either 0 or 1. Determines if an additional "+1" is added in the numerator of intermediate p-value computation. Default: 1.
- `method` is the FDR controlling method in the last step. Either `"BH"` (default) or `"BY"`.
- `pvals` if the aggregated p-values should be reported (logical). Default: `FALSE`.

If the function is executed with all default values, the pKO filter is performed with the authors' recommended parameters. It returns the aggregated selection set $\hat{\mathcal{S}}_{pKO}$, the number of knockoff matrices B and (if specified) the vector of aggregated p-values. We illustrate `run.pKO` with $B = 10$ knockoff draws and the same data example as above.

```
res.pKO <- run.pKO(X, y, B = 10, pvals = TRUE)
res.pKO
$Shat
[1]   8   9  27  48  49  55  58  72  73  91  97 108 115 119 121
 129 139 142 144 146 147 151 155 158
[25] 167 171 174 181 186

$B
[1] 10

$pvals
[1] 0.12833333 1.00000000 1.00000000 0.41666667 1.00000000
1.00000000 1.00000000 0.01666667
[9] 0.01666667 0.37833333 1.00000000 1.00000000 1.00000000
0.44500000 1.00000000 0.11166667
[17] 1.00000000 1.00000000 1.00000000 0.40666667 1.00000000
1.00000000 1.00000000 1.00000000
...
```

Note that we have only displayed the first 24 aggregated p-values (and not all 200) due to space.

ADAGES

The multiple knockoff construction and the aggregation with ADAGES can be carried out at once by

```
run.ADAGES(X, y, knockoffs = create.second_order,  statistic = stat.glmnet_coefdiff,
           q = 0.2, K = 5, offset = 1, type = "ADAGES", sets = FALSE)
```

where the user can customize the following settings:

- `X` the $n \times p$ design matrix and `y` and the $n \times 1$ response vector.
- `knockoffs` is the function for the knockoff construction. It must take the $n \times p$ data matrix as input and it must return a $n \times p$ knockoff matrix. The user can either choose a knockoff sampler of the `knockoff` package or define it manually. Default: `create.second_order`.
- `statistic` is a function that computes the scores W_j. It must take the data matrix, knockoff matrix and response vector as input and it outputs a vector of computed scores. The user can either choose one score statistic from the `knockoff` package or define it manually. Default: `stat.glmnet_coefdiff`.
- `q` defines the nominal level for the FDR control. Default: 0.2.
- `K` is the number of knockoff runs. Default: 5.
- `offset` either 0 (knockoff) or 1 (knockoff+). Default: 1.
- `type` either `ADAGES` (default) or `ADAGES.mod` (see below).
- `sets` logical argument if the K selection sets of each knockoff run before the aggregation should be returned. Default: `FALSE`.

The user can choose between the two criteria for the determination of c by modifying the argument `type`. The default option `ADAGES` refers to the ADAGES procedure that minimizes the complexity ratio criterion for the determination of the threshold c^*

$$c^* = \arg\min\{\eta_c : 1 \leq c \leq c_0,\ |\hat{\mathcal{S}}_{(c+1)}| > 0\}, \quad \eta_c = \begin{cases} \dfrac{|\hat{\mathcal{S}}_{(c)}|}{|\hat{\mathcal{S}}_{(c+1)}|}, & |\hat{\mathcal{S}}_{(c+1)}| > 0 \\ \infty, & |\hat{\mathcal{S}}_{(c+1)}| = 0, \end{cases}$$

whereas `ADAGES.mod` applies the threshold-complexity trade-off criterion

$$c^* = \underset{1 \leq c \leq c_0}{\arg\min}\, c|\hat{\mathcal{S}}_{(c)}|.$$

For more information about the adaptive threshold determination, we refer to the original paper of ADAGES by Gui (2020).
The function `run.ADAGES` returns the aggregated selection set $\hat{\mathcal{S}}_{(c^*)}$, the optimal threshold c^*, the number of knockoff runs K and (if specified) the individual selection sets of each knockoff run. We continue with code showing the execution of ADAGES with default values on our exemplary data and its output.

```
res.ADAGES <- run.ADAGES(X, y, sets = TRUE)
> res.ADAGES
$Shat
 [1]   8   9  16  27  48  49  55  58  72  73  80  91  97 108 115
 119 121 126 129 139 142 144 146 147
[25] 151 155 158 163 165 167 171 181 186

$c
[1] 2

$sets
$sets$S1
```

```
[1]   8   9  27  48  49  55  58  72  73  91  97 108 115 119 121
 126 129 139 142 144 146 147 151 155
[25] 158 167 171 181 186

$sets$S2
[1]   8   9  16  27  48  49  55  58  72  73  80  83  91  97 108
115 119 121 129 133 139 142 144 146
[25] 147 151 155 158 160 163 165 167 171 181 186

$sets$S3
[1]   8   9  27  48  49  55  58  72  73  80  91  97 108 115 119
 121 129 139 142 144 146 147 151 155
[25] 158 165 167 169 171 181 186

$sets$S4
[1]   8   9  16  27  48  49  55  58  72  73  91  97 108 115 119
 121 126 129 139 142 143 144 146 147
[25] 151 155 158 163 165 167 171 181 186

$sets$S5
[1]   8   9  27  48  49  55  58  72  73  91  97 108 115 119 121
 129 139 142 144 146 147 151 155 158
[25] 163 167 171 181 186
```

In this example, the optimal threshold is $c^* = 2$. Hence, the aggregated selection set is not the union but it consists of all variables that occur at least two times across the individual selection sets $\hat{\mathcal{S}}_1, \ldots, \hat{\mathcal{S}}_K$.

Advanced usage

Manual approach

Besides running each procedure as "all-in-one" approach, the user can also run the steps of one multiple knockoff filter manually which provides more flexibility. In other words, the user can manually construct multiple knockoff matrices, estimate the K score vectors and determine the aggregated selection set afterwards. The function `multi.knockoffs` takes the $n \times p$ data matrix $\mathbf{X}$, the number of desired knockoff matrices K and a function that samples the knockoffs as input, and it returns a list with K knockoff matrices $\tilde{\mathbf{X}}^{\mathbf{(1)}}, \ldots, \tilde{\mathbf{X}}^{\mathbf{(K)}}$.

```
mult.knockoffs(X, K, knockoffs = create.second_order)
```

The argument `knockoffs` requires a function taking the $n \times p$ data matrix as input and returning the $n \times p$ knockoff matrix. The default method is the second-order approximation, which is also the default in the `knockoff` package. The user can also choose `create.gaussian` from the `knockoff` package or any other user-defined knockoff sampler.
Once we have drawn the list of K knockoff matrices, we can use them to determine the score vectors $\mathbf{W}^{(k)}$ based on the data $([\mathbf{X}\ \tilde{\mathbf{X}}^{\mathbf{(k)}}], y)\ \forall k$ and the corresponding selection sets $\{\hat{\mathcal{S}}_k : k = 1, \ldots, K\}$. More precisely, we can apply

```
mult.knockfilter(X, Xk, y, q = 0.2, offset = 1, statistic = stat.glmnet_coefdiff)
```

with the arguments:

- `X` the $n \times p$ design matrix.
- `Xk` a list with K elements containing the $n \times p$ knockoff matrices.
- `y` the $n \times 1$ response vector.

- `q` either a scalar or vector of nominal levels. If a scalar is supplied, then the same nominal level is used for each knockoff run. Default: 0.2.
- `offset` either 0 (knockoff) or 1 (knockoff+). Default: 1.
- `statistic` the function to compute the score statistics. The user can either choose any implemented score function of the `knockoff` package or define one by herself. If manually defined, the score function has to take the matrices $\mathbf{X}$, $\tilde{\mathbf{X}}$ and the vector y as input and must return a vector of score statistics $\mathbf{W}$. Default: LCD score with a λ tuned by CV `stat.glmnet_coefdiff`.

The function outputs a list containing three elements:

- `W.list` a list containing the K score vectors of each knockoff run.
- `Shat.list` a list containing the K selection sets of each knockoff run.
- `q` a vector containing the nominal levels of each knockoff run.

Both functions, `multi.knockoffs` and `multi.knockfilter`, are especially useful if the user wants to conduct simulations comparing the implemented aggregation schemes, or if she wants to investigate a certain step of the multiple knockoff filter estimation more thoroughly.
With the score vectors and selection sets determined, the three presented aggregation methods can be used to obtain the aggregated selection set. The function

```
agg.union(Shat.list)
```

takes a list of K selection sets as input and outputs the union of them. The idea of taking the union of selection sets can generally be applied to aggregate any selection procedures and is not limited to the knockoff framework. The function just requires that the selection sets are stored in a list. The user can also run the three so far presented functions to carry out the union knockoff steps manually by defining the sequence of nominal levels q_k for each knockoff run, that is $q_k = 0.2/2^{k-1}$ in the example below.

```
K <- 5
Xk <- mult.knockoffs(X, K = K)
#Xie and Lederer's recommended sequence
q <- 0.2/ 2^((1:K)-1)
knock.res <- mult.knockfilter(X, Xk, y, q = q)
UKO.res <- agg.union(knock.res$Shat.list)
```

We continue with the explanation of `agg.pKO`. Although this command does not have much practical benefit, since pKO can only be applied in the knockoff framework, it is useful in simulations, or if the user wants to investigate the behaviour of the aggregated p-values with different options.

```
agg.pKO(W.list, q = 0.2, gamma = 0.3, offset = 1, method = "BH",  pvals = FALSE)
```

It takes similar arguments as `run.pKO`:

- `W.list` a list with B elements containing the vectors of scores with length p each.
- `q` defines the nominal level for the FDR control. Default: 0.2.
- `gamma` is a value between $(0, 1)$ which defines the quantile value used for the aggregation. If `gamma = NULL`, the adaptive search by Meinshausen et al. (2009) is used. Default: 0.3.
- `offset` either 0 or 1. Determines if an additional "+1" is added in the numerator of intermediate p-value computation. Default: 1.
- `method` is the FDR control method in the last step. Either `"BH"` (default) or `"BY"`.
- `pvals` if the aggregated p-values should be reported (logical). Default: `FALSE`.

The command returns the aggregated selection set, the number of knockoff draws B and (if specified) the aggregated p-values. Similar to the example above, the user can run each step of the pKO procedure manually.

```
K <- 5
Xk <- mult.knockoffs(X, K = K)
knock.res <- mult.knockfilter(X, Xk, y)
pKO.res <- agg.pKO(knock.res$W.list)
```

Note that for the manual execution of pKO, the scalar or vector of nominal levels in `multi.knockfilter` can be arbitrary since we only need the list of score vectors whose computation does not depend on the nominal level.
The last aggregation scheme ADAGES can be performed by

```
agg.ADAGES(Shat.list, p = p)
```

which takes a list of K selection sets and the number of variables p of the model as input, and it returns a list with the aggregated set, the optimal threshold c^* and the number of runs knockoff runs K. Similar to `agg.union`, `agg.ADAGES` is not restricted to the knockoff framework, and it can generally be applied to aggregate multiple runs of variable selection procedures or even different variable selection procedures that have FDR control at q respectively. To run ADAGES manually in the multiple knockoff framework, the user can execute the code below.

```
K <- 5
Xk <- mult.knockoffs(X, K = K)
knock.res <- mult.knockfilter(X, Xk, y)
ADAGES.res <- agg.ADAGES(knock.res$Shat.list, p = p)
```

Change of the optimization technique of knockoff samplers

The (A)SDP optimization is used by default when applying the second-order construction in `multi.knockoffs`, `run.uKO`, `run.pKO` and `run.ADAGES`. The user can modify the knockoff construction within our functions, e.g. changing the optimization from (A)SDP to equi-correlation. We illustrate this with `multi.knockoffs`, but it works equivalently for the other functions.

```
equi.knock<- function(X) create.second_order(X, method = "equi")
Xk <- mult.knockoffs(X, K = K, knockoffs = equi.knock)
```

References

Barber, R. F. and E. J. Candès (2015). Controlling the false discovery rate via knockoffs. The Annals of Statistics 43(5), 2055-2085.

Candès, E., Y. Fan, L. Janson, and J. Lv (2018). Panning for gold: 'model-X' knockoffs for high dimensional controlled variable selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 80(3), 551-577.

Gui, Y. (2020). ADAGES. Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference. ACM.

Meinshausen, N., L. Meier, and P. Bühlmann (2009). p-Values for High-Dimensional Regression. Journal of the American Statistical Association 104(488), 1671-1681.

Nguyen, T.-B., J.-A. Chevalier, B. Thirion, and S. Arlot (2020). Aggregation of Multiple Knockoffs. Proceedings of the 37th International Conference on Machine Learning. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 7283-7293.

Patterson, E. and M. Sesia (2020). knockoffs: The Knockoff Filter for Controlled Variable Selection. R package version 0.3.3. https://CRAN.R-project.org/package=knockoff

Xie, F. and J. Lederer (2021). Aggregating Knockoffs for False Discovery Rate Control with an Application to Gut Microbiome Data. Entropy 23(2), 230.