Using the cheapknockoff package

Guo Yu

2020-04-12

The `cheapknockoff` package contains the implementations of a procedure for performing feature selection when features have costs. The basic premise is to construct multiple knockoffs for each feature, forcing the more expensive feature to compete with more knockoffs. This makes the more expensive feature harder to be selected unless it shows significant association with the response given its cost. The details of our proposal can be found in Yu, Witten, and Bien (2019) *Controlling Costs: Feature Selection on a Budget*.

Data simulation

We consider the following linear model,

$$y = \sum_{j=1}^{p} X_j \beta_j^* + \varepsilon,$$

where $X \sim N(\mathbf{0}, \mathbf{I}_p)$, and $\varepsilon \sim N(0, 1)$. The following code simulates $n = 100$ observation from the model above with $p = 30$.

```
library(cheapknockoff)
set.seed(123)
n <- 100
p <- 30
x <- matrix(data = rnorm(n * p), nrow = n, ncol = p)
y <- x[, 1] - 2 * x[, 2] + rnorm(n)
```

Additionally, we consider the setting where these features have costs. For example, we set the costs for the two relevant features X_1 and X_2 as 2 and 9, and the costs for other features are randomly selected integer values between 2 and 9.

```
omega <- c(2, 9, sample(seq(2, 9), size = 28, replace = TRUE))
```

Using `cheapknockoff` functions

Performing the cost-conscious feature selection procedure using `cheapknockoff` package involves the following three steps:

1. Construct multiple knockoffs
2. Compute knockoff statistics
3. Output the path of selected variables

Constructing multiple knockoffs

The function `multiple_knockoff_Gaussian` constructs multiple knockoffs by assuming that the design matrix follows a multivariate Gaussian distribution with known mean and covariance matrix (which, in this simulation setting, are $\mathbf{0}$ and $\mathbf{I}$ respectively). It also requires the input of feature costs.

```
X_k <- multiple_knockoff_Gaussian(X = x, mu = rep(0, p), Sigma = diag(1, p), omega = omega)
```

Constructing multiple knockoffs involves solving a optimization problem. There are three formulations of the involved optimization problem, which can be specified as an input in `knockoff_Guassian`. For details, see

Gimenez & Zhou (2018) *Improving the Stability of the Knockoff Procedure: Multiple Simultaneous Knockoffs and Entropy Maximization*.

Computing knockoff statistics

With the constructed multiple knockoffs, the next step is to compute the test statistics used to perform feature selection. In particular, the function `stat_glmnet_coef` computes the statistics based on the absolute value of the coefficient estimate from `glmnet` fit. For further details, see Section 2.2 of Yu, Witten, and Bien (2019) *Controlling Costs: Feature Selection on a Budget*.

```
stat <- cheapknockoff::stat_glmnet_coef(X = x, X_k = X_k, y = y, omega = omega)
```

The output of `stat_glmnet_coef` is a list of three components. `kappa` is the vector of indices of `winner` for each variable competing with its multiple knockoff counterparts. `kappa[j] = 1` indicates that the original variable is beating all of its knockoff counterparts, and `kappa[j]` not equal to `1` means otherwise. `tau` is a vector of scores determining the order for which we consider to include variables into the model. Finally, `score_total` is the original `glmnet` coefficient estimates for each variable and its knockoff counterparts.

Building the path of selected variables

Finally, the following function `generate_path` yields a path of selected variables.

```
path <- cheapknockoff::generate_path(kappa = stat$kappa, tau = stat$tau)
```

The output `path` is a list of selected variable sets. For example,

```
path[1]
#> [[1]]
#> [1] 1
```

is X_1. The more relavant feature X_2 (in the sense that X_2 has larger coefficient) is later included in

```
path[2]
#> [[1]]
#> [1] 1 2
```

It is because X_2 is much more expensive than X_1. This relects the premise of our proposal on making use of cheap features over expensive features.