Using the Caseinflu package

Qiuyu Gu

2022-09-15

The `Caseinflu` package contains the procedure of case influence assessment, which is based on case influence graph (Cook, R. D. (1986). *Assessment of local influence.*). For each observation i^*, the case influence graph is a function of case weight parameter w. At each $w > 0$, the function value is the "difference" between the full-data model and the case weight adjusted model, which can be formalized as

$$\text{Minimize}_\theta \ wL_{i^*}(\theta) + \sum_{i \neq i^*} L_j(\theta) + \frac{\lambda}{2}\|\theta\|_2^2 \tag{1}$$

if we consider the L_2-regularized M-estimation problem. Here the "difference" can be quantified as the sum of the square of the fitted values if we consider the regression models. As for the classification models, we can measure the "difference" by comparing the fitted margins under full-data model and case weighted adjusted model. For more details, see section 5.1.1 of Tu, S. (2019). *Case Influence and Model Complexity in Regression and Classification*.

To obtain the case influence graph for each case, we need to solve (1) for every $w > 0$, which can be view as a solution path problem. We proposed two path-following algorithms to solve (1) efficiently: exact path-following algorithm for nonsmooth loss function $L_i(\theta)$, including quantile regression and support vector machine, and approximate path-following algorithm for smooth loss function, including common generalized linear models. **The details of the proposed path-following algorithms can be found in .....** .

Using `Caseinflu` functions

For each given data and specified model, `Caseinflu` package can generate three different case influence assessment metrics:

1. Global influence (area under case influence graph)

2. Cook's distance

3. Local influence. For more details, see section 5.1.3 of Tu, S. (2019). *Case Influence and Model Complexity in Regression and Classification*.

Next, we use the following examples to illustrate how we can obtain theses metrics.

Nonsmooth models

We use quantile regression as an example to show how to apply `Caseinflu` functions to do case influence assessment under nonsmooth models. First we generate data from the standard linear model

$$y_i = x_i^\top \beta + \epsilon_i,$$

where covariates $\{x_{ij} : i = 1, \cdots, n,\ j = 1, \cdots, p\}$, coefficients $\{\beta_1, \cdots, \beta_p]\}$ and errors $\{\epsilon_i : i = 1, \cdots, n\}$ are independently generated from standard normal distribution. Here tau is the parameter in quantile regression and lam is the regularization parameter for L_2 penalty.

```
# n = 100
# p = 200
# lam = 50
# tau = 0.1
# beta_true = rnorm(p)
# X = matrix(0, ncol = p, nrow = n)
# for (i in 1:n)
#   X[i, ] = rnorm(p)
# eps = rnorm(n)
# Y = X %*% beta_true + eps
```

To compute **global influence** for the generated data under quantile regression, we can call the function `Compute_CaseInflu_nonsmooth`. All we need to do is to specify the `class = "quantile"`. The function will output a vector of length n that is the global influence for each case. To get the global influence, the function will compute the path for (1) as w changes from 1 to 0, so the LOO solution can be directly obtained without extra computations. Then the function will also output a vector of length n that is the **Cook's distance** for each case.

```
# case_influence_quantile = Compute_CaseInflu_nonsmooth(X, Y, lam, class = "quantile", tau = tau)
# case_influence_quantile$global_influence
# case_influence_quantile$cook_distance
```

To make it work for support vector machine (svm), the following changes are needed:

1. Set `class = "svm"`
2. Do not need to specify values for tau
3. Specify the values for argument `influence_measure`, since svm is a classification model. Two options for argument `influence_measure` are "BDLD" and "FMD".

```
#case_influence_svm = Compute_CaseInflu_nonsmooth(X, Y, lam, class = "svm", influence_measure = "BDLD")
```

To compute **local influence**, which only depends on the full-data solution, for the generated data under quantile regression, we can call the function `Compute_LocalInflu_nonsmooth`. It will output a vector of length n that is the **local influence** for each case.

```
# local_infuence_quantile = Compute_LocalInflu_nonsmooth(X, Y, lam, class = "quantile", tau = tau)
```

In order to compute the **local influence** for svm, we need to make the same adjustments as what we did for function `Compute_CaseInflu_nonsmooth`.

```
# local_infuence_svm = Compute_LocalInflu_nonsmooth(X, Y, lam, class = "svm", influence_measure = "BDLD
```

Smooth models

We use logistic regression as an example to show how to apply `Caseinflu` functions to do case influence assessment under smooth models. First we generate data as follows:

1. Generate each component of response vector $Y \in \mathbb{R}^n$ independently from Bernoulli distribution, where $\mathbb{P}(Y_i = -1) = \mathbb{P}(Y_i = 1) = 0.5$.

2. Conditional on Y_i, the covariate x_i is generated from $N(Y_i\mu, \sigma^2 I_{p\times p})$ with $\mu = (1/\sqrt{p}, \cdots, 1/\sqrt{p})$ and $\sigma = 1$.

```
# n = 100
# p = 200
# mu = rep(1/sqrt(p),p)
# sigma = 1
# Y = rep(-1, n)
# Y[runif(n) > 0.5] = 1
# X = matrix(0, ncol = p, nrow = n)
# for (i in 1:n)
# {
#   X[i, ] = mvrnorm(n = 1, mu * Y[i], Sigma = sigma*diag(p))
# }
```

To compute **global influence** for the generated data under logistic regression, we can call the function `Compute_CaseInflu_GLM`. Since the path-following algorithm to solve (1) under smooth models is different from the one under nonsmooth models, we have some extra arguments in function `Compute_CaseInflu_GLM`.

1. `epsilon`: the specified error tolerance for the path-following algorithm.

2. `t_max`: we reparametrized w as e^{-t} for smooth models, and t_{max} indicates that we will approximate the solution path for (1) over $w \in [e^{-t_{max}}, 1]$. The default value for `t_max` is 10.

3. `method`: we will need to solve the problem (1) at each selected grid point in the path-following algorithm, and `method` argument specifies which optimization algorithm to compute the solution at each grid point. Two options are Newton method (`method = "Newton"`) and gradient descent (`method = "GD"`). The default value is "Newton".

Here is the command to compute `global influence` and `Cook's distance` under logistic regression.

```
# lam = 1
# eps = 1e-4
# t_max = 10
# CaseInfluence = Compute_CaseInflu_GLM(X, Y, lam, eps, t_max, class = "logistic", method = "GD", influ
# case_influence_quantile$global_influence
# case_influence_quantile$cook_distance
```

For the computation of global influence under other generalized linear models, we only need to specify the argument `class` to the model we want. This version also supports poisson regression (`class = "poisson"`).

To compute **local influence** for the generated data under logistic regression, we can call the function `Compute_LocalInflu_GLM`. It will output a vector of length n that is the **local influence** for each case.

```
# local_influence_logistic = Compute_LocalInflu_GLM(X, Y, lam, class = "logistic", influence_measure =
```

If we change the class as "poisson", then the local influence can be computed for poisson regression.

```
# local_influence_poisson = Compute_LocalInflu_GLM(X, Y, lam, class = "poisson")
```