補論: 推定精度の改善

機械学習

川田恵介 (keisukekawata@iss.u-tokyo.ac.jp)

Table of contents

- R learner の改善には、以下が重要
  - 予測モデルの精度向上
  - 推定に使う事例数の確保
- Stacking + 交差推定 + 多様な推定方法 が重要
  - ddml package で簡単に実装できる

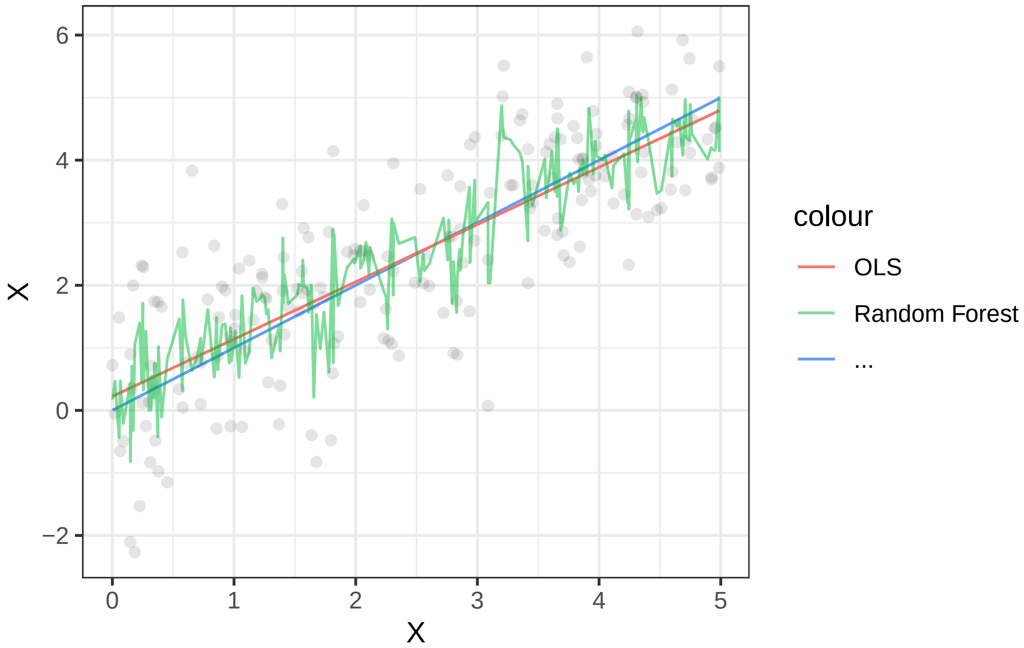
1 Stacking

- OLS や RandomForest 以外にも、大量の推定方法が提案されている
  - どの方法を用いれば良いか?
- 一つの推定方法に絞る必要はなく、複数の予測の加重平均を用いれば良い

1.1 母集団と推定方法

- 一般に、母集団の性質に応じて適した推定方法は異なる
  - 事前にはわからないので、色々試すしかない
- 例: X と Y の関係性が滑らかであれば、決定木系統よりも OLS 系統の方が予測性能が高い傾向

1.2 数値例



1.3 モデル選択/集計

- 多くの推定方法を用いて、さまざまな予測値を推定し、テストデータを用いて
  - 最善の予測値を選択する
  - **予測値の集計を行う** (Stacking)
- Stacking の方が、予測性能が (微妙に) 高い場合が多い

1.4 手順: Short stacking 法

1. 訓練/テストに 2 分割
2. 訓練データを用いて、さまざまな方法で予測モデルを生成
   - OLS, Random Forest など
3. テストデータを用いて、Y を予測値で OLS 回帰し、予測値を以下のように算出する

$$\beta_0 + \beta_1 \times OLSの予測 + \beta_2 \times RandomForestの予測 + ..$$

1.5 実例

```
ModelOLS = lm(Price ~ Size + District,Train)
PredOLS = predict(ModelOLS, Test)

ModelRF = ranger::ranger(Price ~ Size + District,Train)
PredRF = predict(ModelRF, Test)

lm(Price ~ PredOLS + PredRF$predictions, Test)
```

```
Call:
lm(formula = Price ~ PredOLS + PredRF$predictions, data = Test)

Coefficients:
       (Intercept)             PredOLS  PredRF$predictions
          -4.99743             0.05594             1.06825
```

2 交差推定

- 予測値は、予測対象を含まないデータで推定する必要がある
  - 「単純 2 分割」は有力な方法だが、元データの一部しか予測モデルや母集団の特徴の推定に活用できない
  - 交差推定を用いれば、全てのデータを活用できる
    - R-leaarner に応用すれば、信頼区間を狭くできる

2.1 交差推定

0. データを細かく分割 (第 1,..,10 サブグループなど)
1. 第 1 サブグループ**以外**で推定して、第 1 サブグループの予測値を算出
2. 第 2...サブグループについて、繰り返し、全事例に対して予測値を算出

2.2 数値例: 3 分割

```
# A tibble: 9 x 3
```

```
  StationDistance    Price Group
            <int>    <dbl> <fct>
1               9  6.05    3
2               4  3.94    2
3               7 31.0     3
4               1  8.64    1
5               2 -5.99    3
6               7 -4.48    1
7               2 -0.895   1
8               3  0.00785 2
9               1 -3.12    2
```

2.3 数値例: Step 1

```
# A tibble: 9 x 5
  StationDistance    Price Group   OLS RandomForest
            <int>    <dbl> <fct> <dbl>        <dbl>
1               9  6.05    3     NA           NA
2               4  3.94    2     NA           NA
3               7 31.0     3     NA           NA
4               1  8.64    1     -4.12        -1.89
5               2 -5.99    3     NA           NA
6               7 -4.48    1     12.9         16.7
7               2 -0.895   1     -1.29        -1.91
8               3  0.00785 2     NA           NA
9               1 -3.12    2     NA           NA
```

2.4 数値例: Step 2

```
# A tibble: 9 x 5
  StationDistance    Price Group    OLS RandomForest
            <int>    <dbl> <fct>  <dbl>        <dbl>
1               9  6.05    3     NA           NA
2               4  3.94    2      4.86        -0.189
3               7 31.0     3     NA           NA
4               1  8.64    1     -4.12        -1.89
5               2 -5.99    3     NA           NA
6               7 -4.48    1     12.9         16.7
7               2 -0.895   1     -1.29        -1.91
```

```
8                3  0.00785 2      3.55        -0.189
9                1 -3.12    2      0.938        1.91
```

2.5 数値例: Step 3

```
# A tibble: 9 x 5
  StationDistance    Price Group    OLS RandomForest
            <int>    <dbl> <fct>  <dbl>        <dbl>
1               9  6.05    3     -4.88        -1.84
2               4  3.94    2      4.86        -0.189
3               7 31.0     3     -3.03        -1.84
4               1  8.64    1     -4.12        -1.89
5               2 -5.99    3      1.61         0.945
6               7 -4.48    1     12.9         16.7
7               2 -0.895   1     -1.29        -1.91
8               3  0.00785 2      3.55        -0.189
9               1 -3.12    2      0.938        1.91
```

2.6 実例 with ddml package

- 以上の手続きは ddml package で実装可能

```r
library(ddml)
Y = Data$Price
D = Data$D
X = model.matrix(
  ~ 0 + Size + BuildYear,
  data = Data)

Model = ddml_plm(
  y = Y, # Outcome
  D = D, # Treatment
  X = X, # Control
  learners = list(
    list(fun = ols), # Use OLS
    list(fun = mdl_ranger) # Use LASSO
  ),
  shortstack = TRUE, # Use Simple Stacking
  sample_folds = 2 # データを 2 分割
)
```

```
DDML estimation in progress.

E[Y|X]: sample fold 1/2
E[Y|X]: sample fold 2/2 -- Done!

E[D1|X]: sample fold 1/2
E[D1|X]: sample fold 2/2 -- Done!
DDML estimation completed.
```

2.7 実例. 平均差

```r
summary(Model)
```

```
PLM estimation results:

, , nnls

            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.173      0.232  -0.748 4.54e-01
D_r            2.296      0.468   4.906 9.27e-07
```

```r
# 以下でも OK

PsuY = Model$ols_fit$model$y_r
PsuD = Model$ols_fit$model$D_r

estimatr::lm_robust(PsuY ~ PsuD)
```

```
              Estimate Std. Error    t value     Pr(>|t|)   CI Lower  CI Upper
(Intercept) -0.1734108  0.2317008 -0.7484256 4.542321e-01 -0.6276255 0.2808039
PsuD         2.2957283  0.4678192  4.9072985 9.472187e-07  1.3786388 3.2128178
              DF
(Intercept) 6141
PsuD        6141
```

2.8 実例. BLP

- 異質性分析も可能

```r
estimatr::lm_robust(
  PsuY ~ PsuD + PsuD:scale(Data$Size) + PsuD:scale(Data$BuildYear)
```

```
)
```

```
                             Estimate Std. Error    t value     Pr(>|t|)
(Intercept)                -0.1740299  0.2310929 -0.7530734 4.514347e-01
PsuD                        2.2982086  0.4668653  4.9226379 8.761604e-07
PsuD:scale(Data$Size)       2.5588008  0.6889255  3.7141909 2.056640e-04
PsuD:scale(Data$BuildYear)  0.9252957  0.4099547  2.2570683 2.403881e-02
                             CI Lower  CI Upper   DF
(Intercept)                -0.6270529 0.2789931 6139
PsuD                        1.3829891 3.2134281 6139
PsuD:scale(Data$Size)       1.2082654 3.9093362 6139
PsuD:scale(Data$BuildYear)  0.1216409 1.7289506 6139
```

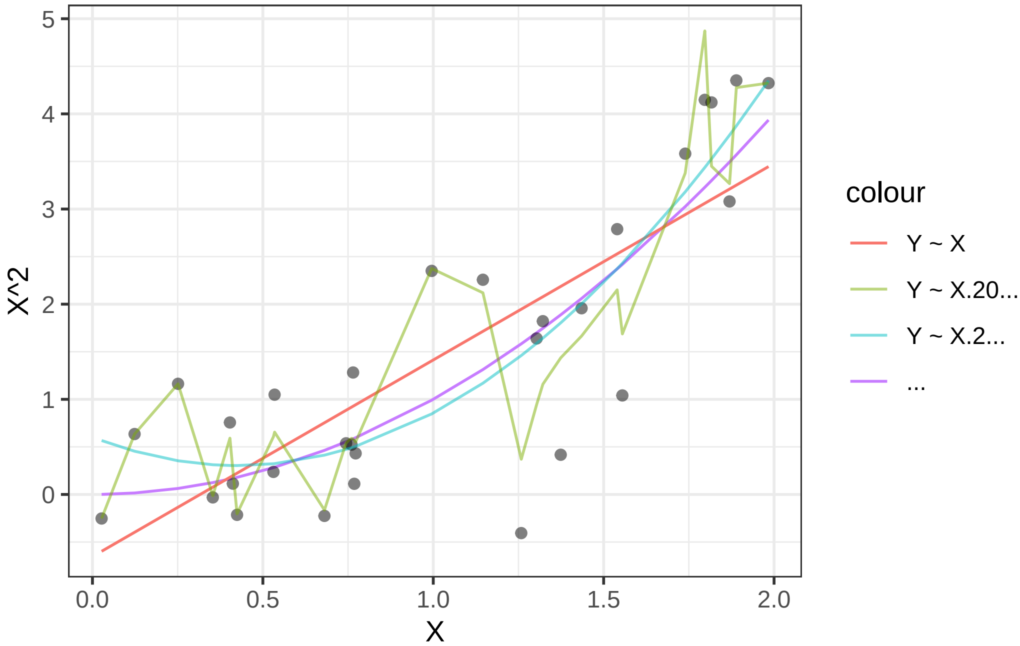
3 Penalized Regression

- OLS は有力な推定方法だが、Model Uncertainly が大きい
  - Model Uncertainly を減らす代表的な方法は LASSO を活用した over-parametrize 推定
  - Stacking の有力な構成要素となりうる

3.1 Recap: OLS

- $g(X) = \beta_0 + \beta_1 X_1 + ..$ と特定化し、訓練データに最も適合するように β を推定
- 非常に柔軟な枠組み: 以下は全て OLS で推定できる
  - $g(Size) = \beta_0 + \beta_1 Size$
  - $g(Size) = \beta_0 + \beta_1 Size + \beta_2 Size^2$

3.2 数値例



3.3 LASSO

0. 十分に複雑なモデルからスタート
1. 何らかの基準 (後述) に基づいて Hyper (Tuning) parameter λ を設定
2. 以下の最適化問題を解いて、Linear model $g(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ...$ を推定

$$\min \sum (y_i - g(x_i))^2 + \lambda(|\beta_1| + |\beta_2| + ..)$$

3.4 Constrained optimization としての書き換え

1. 何らかの基準 (後述) に基づいて Hyper parameter A を設定
2. 以下の最適化問題を解いて、Linear model $g(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ...$ を推定

$$\min \sum (y_i - g(x_i))^2$$

where

$$|\beta_1| + |\beta_2| + .. \leq A$$

3.5 λ の役割: OLS

- $\lambda = 0$ と設定すれば、(複雑なモデルを)OLS で推定した推定結果と一致
  - モデルが複雑すぎる

3.6 λ の役割: 平均

- $\lambda = \infty$ と設定すれば、必ず $\beta_1 = \beta_2 = .. = 0$ となる
  - β_0 のみ、最小二乗法で推定: $g(X) =$ サンプル平均
  - モデルが単純すぎる

3.7 λ の役割

- やりたい事: 予測性能を最大化できるように λ を設定し、単純すぎるモデル (Approximation error が大きすぎる) と複雑すぎるモデル (Estimation error が大きすぎる) の間の" ちょうどいい" モデルを構築する
- 設定方法: サンプル分割 (交差推定, glmnet で実装)、情報基準 (gamlr で採用)、理論値 (hdm で採用)

3.8 実例 with hdm

```r
hdm::rlasso(Y ~ poly(X,10), Temp)
```

```

Call:
rlasso.formula(formula = Y ~ poly(X, 10), data = Temp)

Coefficients:
  (Intercept)   poly(X, 10)1   poly(X, 10)2   poly(X, 10)3   poly(X, 10)4
        1.453          6.571          2.626          0.000          0.000
 poly(X, 10)5   poly(X, 10)6   poly(X, 10)7   poly(X, 10)8   poly(X, 10)9
        0.000          0.000          0.000          0.000          0.000
poly(X, 10)10
        0.000
```

3.9 実例 with ddml package

```
X = model.matrix(
  ~ 0 + poly(Size,2) + poly(BuildYear,2),
  data = Data)

Model = ddml_plm(
  y = Y, # Outcome
  D = D, # Treatment
  X = X, # Control
  learners = list(
    list(fun = ols), # Use OLS
    list(fun = mdl_ranger), # Use LASSO
    list(fun = mdl_glmnet)
  ),
  shortstack = TRUE, # Use Simple Stacking
  sample_folds = 2 # データを 2 分割
)
```

```
DDML estimation in progress.

E[Y|X]: sample fold 1/2
E[Y|X]: sample fold 2/2 -- Done!

E[D1|X]: sample fold 1/2
E[D1|X]: sample fold 2/2 -- Done!
DDML estimation completed.
```

3.10 実例. 平均差

```
summary(Model)
```

```
PLM estimation results:

, , nnls

            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.000451      0.235 0.00191 9.98e-01
```

```
D_r          2.331446       0.473 4.93104 8.18e-07
```

```r
# 以下でも OK

PsuY = Model$ols_fit$model$y_r
PsuD = Model$ols_fit$model$D_r

estimatr::lm_robust(PsuY ~ PsuD)
```

```
                Estimate Std. Error     t value     Pr(>|t|)   CI Lower
(Intercept) 0.0004506655  0.2352987 0.001915292 9.984719e-01 -0.4608172
PsuD        2.3314460871  0.4727327 4.931848357 8.359801e-07  1.4047243
             CI Upper   DF
(Intercept) 0.4617185 6141
PsuD        3.2581678 6141
```

3.11 実例. BLP

- 異質性分析も可能

```r
estimatr::lm_robust(
  PsuY ~ PsuD + PsuD:scale(Data$Size) + PsuD:scale(Data$BuildYear)
  )
```

```
                             Estimate Std. Error    t value     Pr(>|t|)
(Intercept)                0.01273778  0.2350332 0.05419568 9.567810e-01
PsuD                       2.33389533  0.4714355 4.95061462 7.595534e-07
PsuD:scale(Data$Size)      2.80852873  0.6948158 4.04211961 5.361792e-05
PsuD:scale(Data$BuildYear) 0.93032945  0.4136809 2.24890600 2.455379e-02
                             CI Lower  CI Upper   DF
(Intercept)                -0.4480097 0.4734852 6139
PsuD                        1.4097166 3.2580741 6139
PsuD:scale(Data$Size)       1.4464462 4.1706113 6139
PsuD:scale(Data$BuildYear)  0.1193699 1.7412890 6139
```