発展

教師付き学習

川田恵介

Table of contents

様々なアルゴリズム

- 以下を紹介
  - 線形モデル系統: LASSO/Ridge/ElasticNet
  - 決定木系統: Boosting/BART
- 注意: 決定木系統は、少し計算時間が長い

実例: 取引価格予測

```
   nr               task_id                     learner_id resampling_id
1:  1 select(Data, -Reform)                        regr.lm       holdout
2:  2 select(Data, -Reform) mutate.scale.regr.glmnet.tuned       holdout
3:  3 select(Data, -Reform)                     regr.ranger       holdout
4:  4 select(Data, -Reform)              regr.xgboost.tuned       holdout
5:  5 select(Data, -Reform)                       regr.bart       holdout
   iteration  regr.rsq
1:         1 0.8372302
2:         1 0.8504372
3:         1 0.8504834
4:         1 0.8692617
5:         1 0.8660880
Hidden columns: uhash, task, learner, resampling, prediction
```

実例: 改築予測

```
   nr              task_id                     learner_id resampling_id
1:  1 select(Data, -Price)                        regr.lm       holdout
2:  2 select(Data, -Price) mutate.scale.regr.glmnet.tuned       holdout
3:  3 select(Data, -Price)                     regr.ranger       holdout
4:  4 select(Data, -Price)              regr.xgboost.tuned       holdout
5:  5 select(Data, -Price)                       regr.bart       holdout
   iteration  regr.rsq
1:         1 0.1365180
2:         1 0.1406397
3:         1 0.1363812
4:         1 0.1560419
5:         1 0.1593486
Hidden columns: uhash, task, learner, resampling, prediction
```

罰則付き線形モデル

- 線形モデルの OLS 推定は、X が数が多くなると (経験則としてサンプルサイズの 1/3 以上) と性能低下
  - モデルが複雑になりすぎる
  - 複雑性への” 課税”（罰則) が有効

推定方法

- 以下を最小化するように線形モデル $g(X) = \beta_0 + .. + \beta_L X_L$ を推定
- $$\text{モデルへの不適合度(二乗誤差)} + \lambda \times \text{モデルの複雑さ}$$
- λ : HyperParameter (複雑性への課税)
  - 交差推定により決定可能

複雑性の測定

- LASSO: $|\beta_1| + .. + |\beta_L|$
- Ridge: $\beta_1^2 + .. + \beta_L^2$
- Elasticnet:
  - $\alpha[|\beta_1| + .. + |\beta_L|] + (1-\alpha)[\beta_1^2 + .. + \beta_L^2]$
  - α も HyperParameter (交差推定で推定可能)

余談: 良性の過剰適合

- Parameter $>$ 事例数でかつ、$\lambda \to 0$ であれば、データへの適合度は最大 (二乗誤差が 0) になる
  - ただし推定は可能
  - データへの適合度を最大にする制約内で、複雑性が最小になるようにモデルを推定する
- 予測性能が高い場合もある!!!
  - 良性の過剰適合: Spiess, Imbens, and Venugopal (2023)
  - ただしまだまだよくわかっていない現象だそうです

余談: NeuralNet/DeepLearning

- 線形モデルの拡張: 入れ子型のモデルを推定
- NeuralNet
  - $$g(X) = G(\beta_0 + .. + \beta_M \times g_M(X))$$

–

$$g_m(X) = G(\beta_0 + ..\beta_L \times X_L)$$

– $G :=$ リンク関数 (Logit など)

- DeepLearning: 入れ込み構造が多段
  – HyperParameter の調整方法について、膨大な議論が存在

決定木

- 決定木の集計方法に様々な提案

Boosting

- RandomForest: 複雑すぎる決定木の予測を集計し、安定させる (分散を削減する)
- Boosting: 単純すぎる決定木を徐々に複雑化する (ゆっくり学ぶ)
  – 特に伝統的なデータ (非テキスト/画像) に対して、パフォーマンスが高い場合が多いと主張される
  – Einav et al. (2018) でも採用

Boosting: Algorithm

1. 初期予測モデル $g(X) = 0$ を設定し、予測誤差 $r_i = Y_i - g(X) = Y_i$ を計算
2. $r_i \sim X$ を推定 (浅い決定木 $g'(X_i)$ を構築)
3. 予測モデルを更新 $g(X) \leftarrow g(X) + \lambda g'(X)$
4. 予測誤差を更新 $r_i \leftarrow r_i - g(X)$
5. 2-4 を繰り返す

Boosting: HyperParameter

- 個々の木の深さ、繰り返す回数、λ など
- 交差推定で決定可能

Bayesian Additive Regression Tree (BART)

- Irreducible Error が大きい場合にも有効と主張される

- mlr3extralearner (github 経由でインストール可能) に実装
  - 分類問題にも活用可能
- RandomForest: 各決定木を独立して推定
- BART:

BART

- 各更新過程での予測モデル: $f^b(X_i) = \sum_k f_k(X_i)$
- 最終予測モデル: $\sum_{b=L+1}^{B} f^b(X_i)$

BART: Algorithm

1. 初期の予測モデル $g_k^1 = \frac{1}{nK} \sum_i^n Y_i$
2. kth tree の更新: $r_i \sim X$ ただし

$$r_i = Y_i - \sum_{k'<k} f'_{k'}(X_i) - \sum_{k'>k} f_{k'}(X_i)$$

3. 2 を繰り返す

- 各更新は、(1) 新しい分割の追加/選定、(2) 各サブグループの予測値、のどちらかを行う

Reference

Einav, Liran, Amy Finkelstein, Sendhil Mullainathan, and Ziad Obermeyer. 2018. "Predictive Modeling of US Health Care Spending in Late Life." *Science* 360 (6396): 1462–65.

Spiess, Jann, Guido Imbens, and Amar Venugopal. 2023. "Double and Single Descent in Causal Inference with an Application to High-Dimensional Synthetic Control." *arXiv Preprint arXiv:2305.00700*.