AIC vs Early Stopping

Seokhee Moon

October 8, 2020

Contents

This is a documentation for the r code file 'AIC vs ES.r', where the performance of the two different stopping criteria for L2-Boosting algorithm - namely the corrected AIC from [1] and the in-sample Early Stopping - is compared.

1 Load packages

The following packages are required to run the code:

- *mboost*: The function `glmboost` from the *mboost* package is an L2-Boosting algorithm which is used in Section 3.2. The function `mstop` is needed as well to extract the optimal number of boosting iterations based on the corrected AIC in Section 3.4.
- *MASS*: The function `mvrnorm` from the package *MASS* is used to generate a data matrix for normally distributed predictor variables in Section 3.1.
- *gifski*: The function `gifski` from the package *gifski* is used to generate a gif file from the graphical results of the simulation.

2 Basic Set-ups

In this Section the basic parameters are specified, and a data frame is generated to save the simulation results.

First, the basic parameters are set with the following default values:

- `simNUm = 1000`: Number of simulations
- `maxBIter = 600`: Number of the maximum boosting iterations
- `n = 20`: Number of observations
- `p = 10`: Number of the predictor variables
- `s = 2`: Standard error of the error term

Second, an empty data frame `simData` is generated with the following named variables:

- `stopTime_ES`: The optimal number of boosting iterations based on in-sample Early Stopping.
- `mse_ES`: Mean Squared Error (MSE) of the boosting model selected based on in-sample Early Stopping.
- `stopTime_AIC`: The optimal number of boosting iterations based on the corrected AIC.
- `mse_AIC`: MSE (MSE) of the boosting model selected based on the corrected AIC.
- `mse_min`: The minimum MSE value among all boosting iterations.

Each row of the data frame will contain the above numerical results from one simulation.

`simData` :

stopTime_ES	mse_ES	stopTime_AIC	mse_AIC	mse_min
⋮	⋮	⋮	⋮	⋮

3 Simulation

In each simulation, the following steps will be performed.

1. Data generation
2. L2-Boosting - Model training and Performance estimates
3. Model selection via in-sample Early Stopping

4. Model selection via corrected AIC
5. Minimum MSE
6. Plotting

3.1 Generate the data

Here, the basic setting of the simulation in [1] is reproduced.

- `X`: A data matrix of normally and independently distributed predictor variables. And each variable is scaled to have variance of 1.
- `f`: A vector of the true underlying regression values:

  $$f = 1 + 5x_1 + 2x_2 + x_3,$$

  where x_i denotes the i-th predictor variable.
- `eps`: A vector of the error term, which is normally distributed with variace of $\texttt{s}^2 = 4$.
- `Y`: A vector of the observed dependent variable, i.e. the sum of `f` and `eps`.
- `dfTrain`: A data frame of the training data, which contains the named dependent and independent variables (e.g. `Y`, `X1`, `X2`, $\cdots$). This will be used in Section 3.2 to fit L2-Boosting models.

3.2 L2-Boosting

Here, the L2-Boosting models will be trained, and the residual mean squared as well as MSE of each model will be calculated. The residual mean squared is defined as the mean of the squared residuals, i.e. $\frac{1}{n}\sum_{i=1}^{n}(Y - \hat{F})^2$, and it will be used later for the model selection based on in-sample Early stopping in Section 3.3. MSE is the mean of the squared errors, i.e. $\frac{1}{n}\sum_{i=1}^{n}(f - \hat{F})^2$, and is used to evaluate the model performance.

3.2.1 Model training

First, the L2-Boosting models will be trained up to `maxBIter = 600` iterations using the `glmboost` function from the *mboost* package. This result in a glmboost object `L2`, which contains `maxBIter = 600` L2-Boosting models.

```
L2 = glmboost(Y   ., data = dfTrain)
```

3.2.2 Residual mean squared & MSE

First, the following empty vectors are generated to save the Residual mean squared and MSE:

- `resVec`: A vector of length `maxBIter = 600` to save the residual mean squared of the boosting models at each boosting iteration.
- `mseVec`: A vector of length `maxBIter = 600` to save the MSE values of the boosting models at each boosting iteration.

Then for each model (`l2 = L2[iterBoost]`) from `iterBoost = 1` to `600`, the following steps will be repeated:

i Obtain the estimates. $\longrightarrow$ `Fhat`

```
Fhat = predict(l2, newdata = dfTrain[-1])
```

ii Calculate the residual mean squared and save the value in `resVec`.

```
resVec[iterBoost] = mean((dfTrain$Y - Fhat)^2)
```

iii Calculate MSE and save the value in `mseVec`.

```
mseVec[iterBoost] = mean((f - Fhat)^2)
```

3.3 In-sample Early Stopping

The main idea of in-sample Early Stopping criterion is to stop the boosting iteration when the residual mean squared becomes smaller than or equal to the error variance.

Thus it first has to be checked when the residual mean squared first meets the above condition. `idx_ES` is a vector of length `maxBIter = 600`, where it has value of 1 if the condition is satisfied for the given model, and 0 otherwise.

```
idx_ES = ifelse(resVec <= s^2, 1, 0)
```

The optimal number of boosting iterations based on the in-sample Early Stopping is obtained as the index where `idx_ES` has the value 1 for the first time. If `idx_ES` has no value of 1, then the in-sample Early Stopping selects the model at the last boosting iteration `maxBIter = 600`.

```
stopTime_ES = ifelse(sum(idx_ES)>=1, which(idx_ES == 1)[1],
maxBIter)
```

Then the selected model (the optimal number of boosting iterations) as well as its MSE value are saved in `simData`.

```
simData$stopTime_ES[iterSim] = stopTime_ES
simData$mse_ES[iterSim] = mseVec[stopTime_ES]
```

3.4 Corrected AIC

The corrected version of AIC in [1] is already implemented in *mboost* package. The optimal number of boosting iteration based on the corrected AIC can be thus obtained using `mstop` and `AIC` functions for glmboost objects.

```
stopTime_AIC = mstop(AIC(L2, method = 'corrected'))
```

Then the selected model (the optimal number of boosting iterations) as well as its MSE value are saved in `simData`.

```
simData$stopTime_AIC[iterSim] = stopTime_AIC
simData$mse_AIC[iterSim] = mseVec[stopTime_AIC]
```

3.5 Minimum MSE

For each simulation (`iterSim = 1, ..., simNum`), the minimum MSE value among all different boosting models (with the different number of boosting iterations) can be obtained as the minimum value of the vector `mseVec` from Section 3.2.2. And this is saved in the simulation result data frame `simData`.

```
simData$mse_min[iterSim] = min(mseVec)
```

3.6 Plotting

After the above steps from Section 3.1 to 3.5 are completed in each simulation, the performance of all boosting models (`mseVec`) can be plotted as a function of the number of boosting iterations as below in Figure 1. From the graph one can see the general performance of the boosting models as the boosting iteration increases (black solid line), and also easily compare the performance of the two different stopping criteria (red and blue lines).

The graphics generated at the end of each simulation are then exported as png files.

4 Export the results

When all the simulations are done, two files will be exported for the final results:

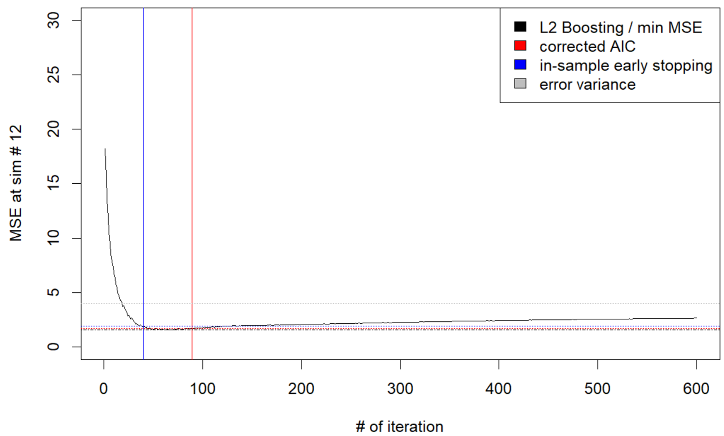

Figure 1: Performance of the Boosting Models

- *'AIC_vs_ES.csv'*: A csv file with all the numerical results from `simData`.
- *'AIC_vs_ES.gif'*: A gif file with all the graphical results from Section 3.6.

References

[1] P. Bühlmann *Boosting for High-dimensional Linear Models*.