STA 663 Final Project

Implementation of Fast FSR Variable Selection

Ran Zhou, Xilin Cecilia Shi

May 1, 2017

1 Abstract

Although variable selection is not widely used in primary analysis of clinical trial data, it is useful in secondary analysis. The goal of this project is to implement the fast false selection rate algorithm and compare it using simulation data with other variable selection algorithms such as LASSO and stepwise selection with BIC. The result indicates that the fast false selection rate method obtains high accuracy in terms of a low false selection rate. Because of the computational simplicity, the intuitive choice of model size and high accuracy, the fast false selection rate is believed to be have a wide application.

Key Words: Model selection, False selection rate, Forward selection, LASSO

2 Background

This project aims to implement the fast false selection rate (Fast FSR) variable selection algorithm proposed in 'Fast FSR Variable Selection with Applications to Clinical Trials' (Boos et al., 2009). The fast FSR variable selection algorithm is a variant of the false selection rate variable selection(FSR) algorithm by Wu, Boos, and Stefanski (2007), henceforth WBS. The FSR by WBS uses simulation by creating phony variables to calculate the estimate of the rate that uninformative variables enter the model, where the simulation is not time efficient. The fast FSR algorithm generates the same net result without creating phony variables, thus the efficiency is highly improved.

Variable selection is an important step for most of the data analysis and model building processes, which is the key to improve prediction accuracy. Reducing the false selection rate can highly save researchers' time. Compared with lasso and BIC, fast FSR algorithm performs better and gives smaller false selection rate no matter when the true number of predictors is large or small. In addition, It costs almost the same amount of time as BIC, but is less efficient than lasso.

Another advantage of Fast FSR is the computational simplicity. When the p-values for the entering variables in the forward selection are monotonically increasing, i.e. $p_1 \leq p_2 \leq \cdots \leq p_{k_T}$, then the implied stopping rule is a version of an adaptive False Discovery Rate (FDR) method, and the model with size k is chosen, where

$$k = max\{i : p_i \leq \frac{\gamma_0(1+i)}{k_T - i}, p_i \leq \alpha_{max}\}$$

3 Algorithm Description

In Fast FSR, the main quantity of interest is the false selection rate given by

$$\gamma = E\{\frac{U(Y,X)}{1 + I(Y,X) + U(Y,X)}\}$$

where $U(Y,X)$ represents the number of uninformative variables in the selected model and $I(Y,X)$ is the number of informative variables. The 1 in the denominator represents the intercept and also avoids problems with dividing by 0. The goal is to adjust the selection method parameters such

that the FSR γ equals to the target rate γ_0. With a specified significance level α, the empirical estimator of γ is given by

$$\gamma = \frac{U(\alpha)}{\{1 + S(\alpha)\}}$$

where $S(\alpha)$ denotes the total number of variables.

The unknown quantity $U(\alpha)$ can be estimated with $\{k_T - S(\alpha)\}\hat{\theta}(\alpha)$, where $\{k_T - S(\alpha)\}$estimates the total number k_U of uninformative variables in the data, and $\hat{\theta}(\alpha)$ estimates the rate that uninformative variables enter the model using tuning parameter α. It is noted that the quantity $\{k_T - S(\alpha)\}$ overestimates the true number of uninformative variables k_U when α is small and underestimates it when α is large. But with an appropriate α, it is a reasonable estimate for sparse models. Putting these together would yield an estimated α as follows,

$$\hat{\alpha} = \sup_{\alpha \leq \alpha_{max}} \{\alpha : \hat{\gamma}(\alpha) \leq \gamma_0\},$$

where

$$\hat{\gamma}(\alpha) = \frac{\{k_T - S(\alpha)\}\hat{\theta}(\alpha)}{1 + S(\alpha)}$$

In forward selection, the p-to-enter refers to the p-value of a variable to be included in the model. The p-to-enter at step i is the smallest p-to-enter of all the variables are that not in the model. When the p-to-enter are monotonically increasing, i.e. $p_1 \leq p_2 \leq \cdots \leq p_{k_T}$, forward selection with significance level α selects a model of size k, where $k = \max\{i : p_i \leq \alpha\}$. When the sequence of p-to-enter are not monotonically increasing, we modify the original sequence by carrying the largest p forward and represent the new monotone sequence of p-to-enter by $\tilde{p_1} \leq \tilde{p_2} \leq p_{\tilde{k_T}}$. Now the forward selection with significance level α selects a model of size k, where $k = \max\{i : \tilde{p_i} \leq \alpha\}$. Under this scenario, at least two of the p-values in the monotone sequence must be equal and at least one of the forward addition sequence is not chosen by forward selection.

Instead of estimating $\hat{\theta}(\alpha)$ via simulation, the Fast FSR method uses $\theta(\alpha) = \alpha$ instead. This is much more desirable since the simulation requires generating equal number of phony variables as the original variables, which is computationally expensive when performing simulation. Then the estimated α becomes

$$\hat{\alpha_F} = \sup_{\alpha \leq \alpha_{max}} \{\alpha : \hat{\gamma_F}(\alpha) \leq \gamma_0\},$$

where

$$\hat{\gamma_F}(\alpha) = \frac{\{k_T - S(\alpha)\}\alpha}{1 + S(\alpha)}$$

This leads to a simple rule for model size selection,

$$k(\gamma_0) = max\{i : \tilde{p_i} \leq \frac{\gamma_0[1 + S(\tilde{p_i})]}{k_T - S(\tilde{p_i})}, \tilde{p_i} \leq \alpha_{max}\}$$

Hence we can obtain $k(\gamma_0)$ by looking at the largest i such that $\tilde{p_i}$ is less than the bound.

The advantage of Fast FSR compared to FSR is that it does not require generating phony variables, hence it is more computationally efficient. WBS recommended permuting the rows of the original X to obtain number of phony variables $k_p = k_T$ (Wu et al., 2007). However, when k_T is large, it requires selecting from $2k_T$ variables each of B times. The computation with the k_T variables may already be heavy, not to mention when including the k_p phony variables.

4 Optimization for Performance

In this section, a few optimization strategies were tried to speed up our algorithm. The strategies include removing redundant calculations, using vectorize computing, and cythonizing the Python code. We have also tried using jit (just-in-time) compiler, but one obstacle we came across was jit and cython cannot be applied to Python code with scikit-learn package which is frequently used in our algorithm. In fact, "linear model" in sklearn is already speeded up by Cython, so the only part we optimized was some other array calculations.

4.1 Remove Redundant Calculation

Removing redundant calculations was the first strategy we used to optimize the code. A lot of calculations were done in separate for loops. To avoid the redundancy, we simply combine the for loops with the same number of iterations.

4.2 Vectorize Computing

List is a convenient data structure for appending elements and removing elements, but the process also wastes a lot of time. When programming, we tried to use numpy array with initialized size instead of keeping appending elements to lists.

4.3 Cythonized Code

Cython was used on some code with a lot of calculations. Helper functions were created in order to speed up part of the code in a function. Two helper functions were cythonized in "fsr.fast.pv" function, which implements Fast FSR based on summary p-values from forward selection.

	Time cost for 3 loops and best of 3
Without Cython	8.7 ms
With Cython	7.7 ms

Table 1: Time Cost Comparison with Cython

As shown in Table 1, we obtained some speed up by rewriting our code in Cython and the time was reduced from 8.7 ms to 7.7 ms after optimizing. The speed up is not very significant and the result is pretty random, which means the original optimization using numpy array is already highly optimized. One reason for having such a small difference might be that Cython will not have a significant improvement on small matrix.

5 Applications to simulated data sets

5.1 Data

We followed the simulation mentioned in the paper for model selection in linear regression. There are two 150 $\times$ 21 design matrices used as input data. The first set was generated independently with $\rho = 0$ from $N(0, 1)$ and the second set was autocorrelated, AR(1) with $\rho = 0.7$. In addition to the 21 original variables, the squared of the 21 variables and their pairwise interactions were also added. Hence the number of total variables is $k_T = 252$. The added variables all have true coefficients equal to 0. The results for $k_T = 21$, 42 and 252 are quite similar and we report only the case of $k_T = 21$ in this paper.

The responses were generated from 5 models with the following labels: H0: all βs = 0; H1: β_7=1, $\beta_1$4=1; H2: $\beta_6$6=9, β_7=4, β_8=1, $\beta_1$3=9, $\beta_1$4=4, $\beta_1$5=1; H3: β_5=25, β_6=16, β_7=9, β_8=4, β_9=1, β_{12}=25, β_{13}=16, β_{14}=9, β_{15}=4, β_{16}=1; H4: β_4=49, β_5 = 36, β_6=25, β_7=16, β_8=9, β_9=4, β_{10}=1, β_{11}=49,β_{12}=36, β_{13}=25, β_{14}=16, β_{15}=9, β_{16}=4, β_{17}=1. The coefficients were standardized to achieve a theoretical R^2=0.35. With the known 'true model', we can test the accuracy of the models on this simulated data set. The authors posted their simulated data sets publicly online and we used the same data set to check our results.

In addition to Fast FSR, we also ran BIC and LASSO on the data sets and compared the results of the three models. In the originally paper, the authors used the R package leaps for best subset selection. However, there does not exist corresponding package or function in Python and hence we also implemented a regression subset selection. The other method LASSO and cross validation was computed with the package scikit-learn.

5.2 Model Comparison

We used several measures to compare the performance among Fast FSR, BIC, and LASSO. These measures include: (1) size: average model size. (2)ME: average model error $= (1/n)|\hat{Y} - \mu|^2$. (3)FSR: average false selection rate = average of (number of unimportant variables selected)/(1+ number of total variables selected), and a lower value is preferred. (4) MSE: average error mean square of the chosen model and close to σ^2 suggests selection method is tuned. (5) CSR: the average proportion of correctly chosen variables, and a lower value is preferred. (6) JAC: Jaccard's measure that combines FSR and CSR in a particular way, and a lower value is preferred. The Monte Carlo standard errors of the above measures were calculated as well.

Figure 1 to Figure 4 show the results of the three models in terms of FSR, CSR and model size. As can be seen, Fast FSR leads to a small and stable FSR across all 5 models, and is close to the advertised 0.05 FSR. The other two methods BIC and LASSO have much larger FSR. It is also noted that the FSR for BIC and LASSO decreases as the number of nonzero βs in the model increases. CSR is complementary to FSR, and together with the model size provides a full picture of the model selection characteristics of the each method.

It can be seen that LASSO has a much higher FSR and CSR compared to Fast FSR and BIC. The reason is that LASSO selects too many variables, which is shown in the plot of model size. LASSO uses shrinkage to shrink the coefficients to 0 and the amount of shrinkage depends on λ, which is determined by cross validation.

The simulations suggest that Fast FSR has the closest FSR with the advertise 0.05 FSR rate. In addition, Fast FSR chooses smaller models and results in lower FSR and CSR.

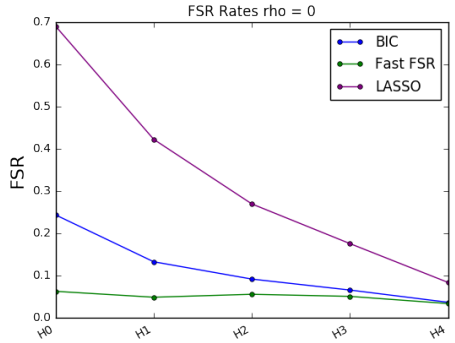

Figure 1: False selection rate.

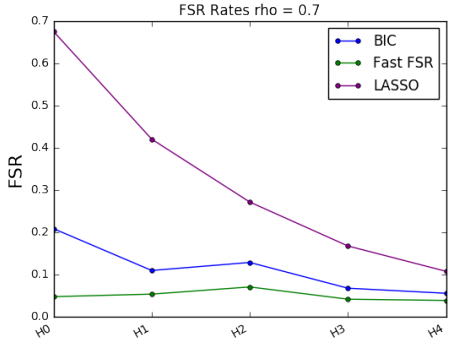

Figure 2: False selection rate.

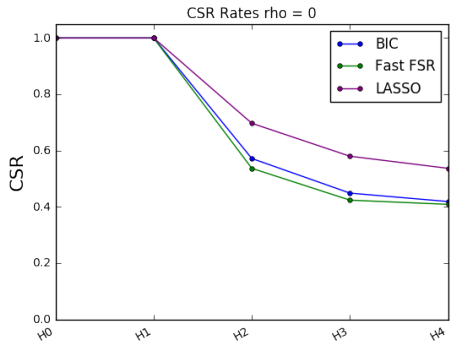

Figure 3: Correct selection rate.

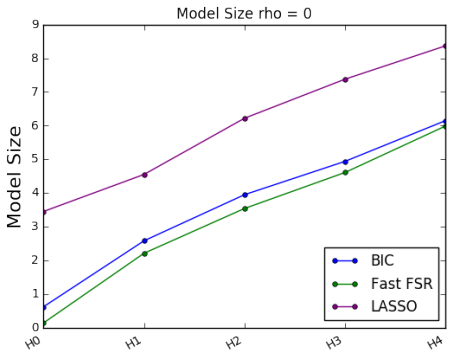

Figure 4: Average model size.

6 Applications to Real Data Sets

We also tested Fast FSR on real-word data sets and this section also presents how the algorithm works in a closer detail. The example that we used is the NCAA Data on 6 year college graduation rates. Table 2 shows how forward selection chooses variables step by step. As what is described in

the algorithm section, Fast FSR chooses the model size k by

$$k = \max\{i : \tilde{p}_i \leq \alpha\},$$

where α is estimated by

$$\hat{\alpha_F} = \sup_{\alpha \leq \alpha_{max}} \{\alpha : \hat{\gamma_F}(\alpha) \leq \gamma_0\}.$$

and

$$\hat{\gamma_F}(\alpha) = \frac{\{k_T - S(\alpha)\}\alpha}{1 + S(\alpha)}$$

This leads to a simple rule for model size selection,

$$k(\gamma_0) = max\{i : \tilde{p}_i \leq \frac{\gamma_0[1 + S(\tilde{p_i})]}{k_T - S(\tilde{p_i})}, \tilde{p}_i \leq \alpha_{max}\}$$

Table 2 shows how to use Fast FSR to select the desirable model. By calculation the estimated α is $\hat{\alpha_F} = 0.0214$. we see that $\tilde{p_5}$ is less than the bound $\hat{\alpha_F}$ but $\tilde{p}$ values after that are not less than the bound. The cutoff line in Table 2 shows where the Fast FSR stops at in forward selection. Fast FSR selects a model size $k(0.05) = 5$ when $\gamma_0 = 0.05$ with variables x_2, x_{12}, x_5, x_4, and x_7 with .

The table contains a column of 'Mono $\tilde{p}$', which represents the monotonized p-values. When the true p-to-enter is not monotonically increasing, we monotonize the original sequence by carrying the largest p forward. Under this situation, at least two of the monotonized p-values must be equal, and there will be a gap in the model size. For example in step 4, the true p-to-enter is 0.0053, which is smaller than the p-to-enter in the step 3. Hence we carry the p-to-enter from step 3 in step 4 and the same case also happens in step 5. As a result, 'x_5', 'x_4' and 'x_7' always need to be included together. Therefore the model size jumps to 5 after step 2 and it is impossible to have model of size 3 or 4.

Step	Variable	p-to-enter	Mono $\tilde{p}$	Size
1	x_2	1.11e-16	1.11e-16	1
2	x_{12}	6.95e-05	6.95e-05	2
3	x_5	0.0115	0.0115	5
4	x_4	0.0053	0.0115	
5	x_7	0.0025	0.0115	
6	x_{17}	0.0433	0.0433	6
7	x_{15}	0.0527	0.0527	7
8	x_6	0.1056	0.1056	10
9	x_9	0.0826	0.1056	
10	x_8	0.0536	0.1056	
11	x_{12}	0.2350	0.2350	11
12	x_{10}	0.2864	0.2864	12
13	x_{13}	0.3163	0.3163	14
14	x_{18}	0.2697	0.3163	14
15	x_{11}	0.4953	0.4953	15
16	x_1	0.6326	0.6326	16
17	x_{14}	0.7056	0.7056	17
18	x_{19}	0.8605	0.8605	18
19	x_{16}	0.9032	0.9032	19

Table 2: Forward Selection

Figure 5 shows the relationship between $\hat{\gamma_F}$ and $\hat{\alpha_F}$. It illustrates: $\gamma_0 = 0.07$ chooses $\hat{\alpha_F} = 0.01$, which gives a model size of 5, and $\gamma_0 = 0.16$ chooses $\hat{\alpha_F} = 0.1$, which gives a model size of 7.

This example shows the computational simplicity and intuition of Fast FSR. Once we calculate the monotone p-values, $\gamma_F(\alpha)$ and α_F with a fixed γ_0, the model size is straightforward to obtain.

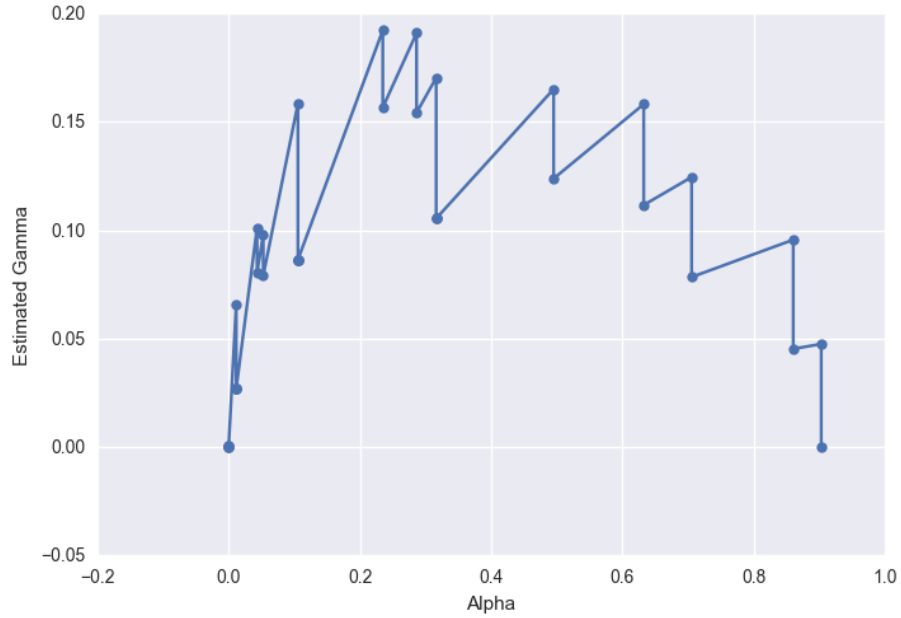

Figure 5: Estimated Gamma for NCAA data set

7 Discussion and Conclusion

The simulation study and experiment on real data sets show that the Fast FSR is a desirable method as it leads to a smaller FSR. However, the authors also mentioned about a limitation of the method that the accuracy decreases when there is high correlation between the informative predictors and noninformative predictors in larger models. Bagging Fast FSR was proposed to address this limitation and to recover good model error performance. Bagging Fast FSR draws multiple bootstrap samples from the original data and uses the average model from each of the bootstrap samples. Such method can reduce the potential drawbacks of forward selection method caused by correlated predictors. We have not yet investigated bagging FSR, and would be an interesting future research to investigate.

Due to the computational simplicity, the intuitive choice of model size and high accuracy, the fast false selection rate is believed to have a wide application. It can be used with almost any regression models, including linear, logistic, and Cox regression. Because of the savings in computations, the Fast FSR method may also be used in permutation tests.

8 Link to Github repository

Our code, examples and data can be found from the following Github repository:

https://github.com/CeciliaShi/STA-663-Final-Project

A detailed description of how to install the package is discussed in the README file in the repository.

9 References

[1] Boos, D. D., Stefanski, L. A., and Wu, Y. (2009). Fast FSR Variable Selection with Applications to Clinical Trials. Biometrics, 65, 692-700.

[2] Boos, D. D., Stefanski, L. A., and Wu, Y. (2007). Controlling variable selection by the addition of pseudovariables. Journal of the American Statistical Association, 102, 235-243.