Feature Selection using Penalized Likelihood function: From optimization for likelihood function to Hyper-parameter tuning

Table

Inwook Back

I. Intro

J. Fan 2001. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties

http://www.personal.psu.edu/ril4/research/penlike.pdf

This article used their own penalized function to estimate beta coefficient and select the variables. What they said good things in their method are those below.

1. Can detect zero beta's well
2. Satisfy three good properties required to a penalized function
   - Unbiasedness
   - Sparsity
   - Continuity
3. Model error is lower than other methods such as Lasso or Best subset.
4. Holds "Oracle property"

So, in this report, I used "Sequential Model Based Optimization" for hyper-parameters tuning.

II. Optimization for Hyper-parameters.

(2.2) is Loss function and (2.7) is the first derivative of penalty function that contains two hyper-parameters(a, lambda)

Denote $\mathbf{z} = \mathbf{X}^T\mathbf{y}$ and let $\hat{\mathbf{y}} = \mathbf{X}\mathbf{X}^T\mathbf{y}$. A form of the penalized least squares is

$$\frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{d} p_j(|\beta_j|) = \frac{1}{2}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 + \frac{1}{2}\sum_{j=1}^{d}(z_j - \beta_j)^2 + \lambda \sum_{j=1}^{d} p_j(|\beta_j|). \quad (2.2)$$

$$p'_\lambda(\theta) = \lambda \left\{ I(\theta \le \lambda) + \frac{(a\lambda - \theta)_+}{(a-1)\lambda} I(\theta > \lambda) \right\}$$
$$\text{for some } a > 2 \text{ and } \theta > 0, \qquad (2.7)$$

Sequential Model Based Optimization)

First, I made a function of hyperparameters to calculate averages of 5-folds CV errors in 100 simulations, and plug in this function to calculate the score value for “SMBO” step.

SMBO(f, M_0, T, S)
1 $\mathcal{H} \leftarrow \emptyset$,
2 For $t \leftarrow 1$ **to** T,
3 $x^* \leftarrow \text{argmin}_x \; S(x, M_{t-1})$,
4 Evaluate $f(x^*)$, ▷ *Expensive step*
5 $\mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$,
6 Fit a new model M_t to $\mathcal{H}$.
7 **return** $\mathcal{H}$

Figure 1: The pseudo-code of generic Sequential Model-Based Optimization.

Run by “Bayesianoptimization” function in “rBayesianoptimization” package.

- Surrogate model : Gaussian Process,
- Acquisition function : GP Upper Confidence Bound.

Advantages of SMBO method

SBMO that searches “a” and “lambda” simultaneously with using good candidates for each iteration, so it would be more reliable than “Random search” and faster than “Grid search”.

III. Optimization for Penalized Loss function.

$$\ell(\boldsymbol{\beta}) + n \sum_{j=1}^{d} p_\lambda(|\beta_j|). \qquad (3.6)$$

(3.6) is a target loss function. But it has singularity at zero. So, the authors approximate this discontinuous non-concave function to the quadratic form (3.7) in order to get second order of derivatives.

$$p_\lambda(|\beta_j|) \approx p_\lambda(|\beta_{j0}|) + \tfrac{1}{2}\{p'_\lambda(|\beta_{j0}|)/|\beta_{j0}|\}(\beta_j^2 - \beta_{j0}^2), \quad \text{for } \beta_j \approx \beta_{j0}. \qquad (3.7)$$

And the target loss function changes into (3.8).

$$\ell(\boldsymbol{\beta}_0) + \nabla\ell(\boldsymbol{\beta}_0)^T(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \tfrac{1}{2}(\boldsymbol{\beta} - \boldsymbol{\beta}_0)^T\nabla^2\ell(\boldsymbol{\beta}_0)(\boldsymbol{\beta} - \boldsymbol{\beta}_0) + \tfrac{1}{2}n\boldsymbol{\beta}^T\Sigma_\lambda(\boldsymbol{\beta}_0)\boldsymbol{\beta}, \qquad (3.8)$$

(3.9) is Newton-Raphson solution of (3.8) target function.

$$\nabla\ell(\boldsymbol{\beta}_0) = \frac{\partial\ell(\boldsymbol{\beta}_0)}{\partial\boldsymbol{\beta}}, \qquad \nabla^2\ell(\boldsymbol{\beta}_0) = \frac{\partial^2\ell(\boldsymbol{\beta}_0)}{\partial\boldsymbol{\beta}\,\partial\boldsymbol{\beta}^T},$$

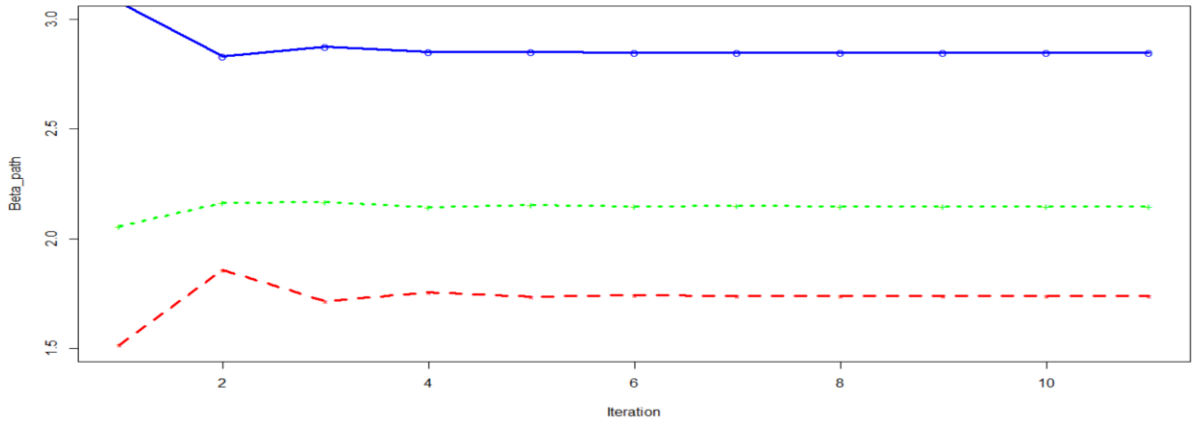
$$\Sigma_\lambda(\boldsymbol{\beta}_0) = \operatorname{diag}\{p'_\lambda(|\beta_{10}|)/|\beta_{10}|, \ldots, p'_\lambda(|\beta_{d0}|)/|\beta_{d0}|\}.$$

The quadratic minimization problem (3.8) yields the solution

$$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 - \{\nabla^2\ell(\boldsymbol{\beta}_0) + n\Sigma_\lambda(\boldsymbol{\beta}_0)\}^{-1}\{\nabla\ell(\boldsymbol{\beta}_0) + n\mathbf{U}_\lambda(\boldsymbol{\beta}_0)\}, \qquad (3.9)$$

where $\mathbf{U}_\lambda(\boldsymbol{\beta}_0) = \Sigma_\lambda(\boldsymbol{\beta}_0)\boldsymbol{\beta}_0$. When the algorithm converges,

I used (3.9) solution for Newton-Raphson algorithm, and the important thing is that U(beta0) matrix should be updated for each iteration. From the 300 times of simulation, the mean of iteration number for convergence is 13.26, so this penalized loss function converges fast.



And this plot shows convergence path of three non-zero beta coefficients. As seen from the third iteration, they are already very close to the values in convergence.

IV. Simulation and Conclusion.

1. Simulation scheme

$$Y = \mathbf{x}^T \boldsymbol{\beta} + \sigma\,\varepsilon,$$

where $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$, and the components of $\mathbf{x}$ and ε are standard normal. The correlation between x_i and x_j is $\rho^{|i-j|}$ with $\rho = .5$. This is a model used in Tibshirani (1996).

- X and Y matrix are generated 100 times for simulation.

2. Hyper-parameters tuning results

Correlation of X = 0.5, # of variables = 20

	Round	a	lambda	Value
1	25	3.705954	0.6548148	-12.42526
2	30	3.698374	0.6392899	-12.85789
3	27	2.000000	0.7587395	-12.91569
4	9	5.125467	0.5362722	-13.27013
5	29	3.474570	0.5301786	-13.48893

Correlation of X = 0.5, # of variables = 8

	Round	a	lambda	Value
1	24	3.655710	0.5588970	-12.82627
2	18	3.663630	0.6058776	-13.04260
3	25	3.279527	0.5674092	-13.04856
4	30	3.798981	0.5007763	-13.43735
5	29	3.743606	0.5712524	-13.52376

Correlation of X = 0.2, # of variables = 20

Correlation of X = 0.2, # of variables = 8

	Round	a	lambda	Value
1	15	3.231404	0.5782307	-12.22391
2	18	2.996700	0.6277029	-12.50059
3	9	3.368765	0.6710693	-12.58052
4	19	3.705172	0.6146233	-12.94791
5	30	4.075337	0.5900784	-12.95939

	Round	a	lambda	Value
1	23	3.220091	0.5645888	-13.35339
2	6	3.771922	0.6240161	-13.86447
3	27	3.636340	0.5287119	-14.16704
4	24	2.588319	0.5278014	-14.18068
5	25	5.901617	0.5914176	-14.21595

The “Value” means “-mean squared error”. And “a” and “lambda” that make this value largest are the optimizing parameters. The first line in each picture are the best hyper parameters.

3. Simulation results.

- n: sample size, d: number of variables, corr(X): correlation between independent variables.
- Fan2001: results in the article Fan(2001),

  Article: results of dataset this time using authors hyper-parameters tuning,

  Mine: results of dataset this time using SMBO tuning.
- MRME(%) is median of 100 times relative model errors.

		Avg. No. of 0 coefficients	Standard deviation of 100 simulations		
Method	MRME(%)	Correct	Beta1	Beta2	Beta3
n=40, d=8, corr(X)=0.5					
Fan2001	47.25	4.29	0.17	0.17	0.15
Article(a=3.70, lambda = 0.65)	65.12	4.18	0.22	0.28	0.27
Mine(a=3.66, lambda = 0.56)	63.64	4.07	0.21	0.26	0.23
Oracle		5.00			
n=60, d=8, corr(X)=0.5					
Fan2001	43.79	4.34			
Article(a=3.70, lambda = 0.65)	62.74	4.28	0.17	0.25	0.22
Mine(a=3.22, lambda = 0.56)	57.97	4.08	0.16	0.20	0.18
Oracle		5.00			
n=150, d=20, corr(X)=0.2					
Article(a=3.70, lambda = 0.65)	80.55	14.98	1.27	0.54	1.27
Mine(a=3.23, lambda = 0.58)	80.00	15.10	1.27	0.58	1.27
Oracle		17.00			

As the difference of hyper-parameters is small, the simulation results are also similar. This result shows that SCAD method can find zero beta coefficients very well.

In conclusion, SCAD method is very fast to convergence and performs very well for the variable selection(Very close to Oracle property).

References

[1] J. Fan 2001. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties

[2] L. Dicker 2013. Variable selection and estimation with the seamless-l0 penalty

[3] JS. Bergstra 2011. Algorithms for Hyper-Parameter Optimization

[4] J. Snoek 2012. Practical Bayesian Optimization of Machine Learning Algorithms