1 the Coordinate descent algorithm implement on square-root lasso

1.1 the square-root lasso and its properties

Considering the linear regression model in :$y_i = x_i^T\beta_0 + \sigma\epsilon$ with independent and identically distributed noise $\epsilon_i \sim F_0$ and $F_0 = \Phi$. The Square-root Lasso proposed by A.BELLON(2011),elimates the need to know or to pre-estimate σ,and can dispense the normality assumption $F_0 = \Phi$. The square-root lasso estimator of β_0 is defined as the solution of the optimization problem as below:

$$\hat{\beta} = argmin_\beta(\Sigma_{i=1}^n(y_i - x_i^t\beta)^2)^{1/2} + \lambda||\beta||_1 \tag{1}$$

Comparing with lasso ,the object function is slightly changed by a square-root,and this give square-root lasso some good properties.The A.Bello(2011)'s article has shown us some good properties of square-root Lasso,and we list them below:

- Achieve the same near-oracle rates of convergence as lasso, without knowing .
- could drop the assumption of noise's normality under specific condition
- the maintenance of global convexity of the objcect function.

1.2 traditional programming method on square-root lasso

In this part we focus on how to solve the square-root lasso by SOCP.
We rewrite the object function (4) the same as it in Bello(2011)'s article:

$$Min\{\hat{Q}(\beta)^{1/2} + \frac{\lambda}{n}\|\beta\|_1\}, \hat{Q}(\beta) = \frac{\Sigma_{i=1}^n(y_i - x_i^t\beta)^2}{n}$$

To minimize the object function,we can do simulation to calculate λ. Then we try to transform it into a conic program problem.we rewrite the object function as:

$$\min_{t,v,\beta^+,\beta^-} \frac{t}{n^{1/2}} + \frac{\lambda}{n}\Sigma_{i=1}^p(\beta_j^+ + \beta_j^-) = (\frac{1}{n^{1/2}}, \frac{\lambda}{n}, ..., \frac{\lambda}{n})(t, \beta_1^+, ..., \beta_p^+, \beta_1^-, ..., \beta_p^-)^t$$

with

$$\beta_j^+ = max(\beta_j, 0), \quad \beta_j^- = -min(\beta_j, 0), \quad \beta = \beta^+ - \beta^-, \quad \beta||_1 = \Sigma_{j=1}^p(\beta_j^+ + \beta_j^-)$$

$$v_i = y_i - x_i^t\beta^+ + x_i^t\beta^-, \quad \hat{Q}(\beta)^{1/2} = \frac{||v||}{n^{1/2}}, \quad Q^{n+1} = \{(v,t) \in R^n \times R : t \geq ||v||\}$$

comparing to the standardized form of second order conic program:

$$\min_u c^t u, ||Au + b||_2 \leq a^t u + d$$

We transform the optimization problem into a SOCP problem and we use package picos in Python to do this job.

1.3 Coordinate descent algorithm on square-root lasso

$$\frac{\partial f(\beta)}{\partial \beta_j} = \frac{\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)(-x_{ij})}{(\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)(-x_{ij})^2)^{1/2}} + \lambda \qquad (\beta_j > 0)$$

$$\frac{\partial f(\beta)}{\partial \beta_j} = \frac{\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)(-x_{ij})}{(\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)(-x_{ij})^2)^{1/2}} - \lambda \qquad (\beta_j < 0)$$

we want

$$a^* = argmin_{y\in R} f(\beta_1, ...\beta_{j-1}, a, \beta_{j+1}, ...\beta_p)$$

let $h(a) = \frac{df(\beta_1,....\beta_{j-1},a,\beta_{j+1},...\beta_p)}{da}$ and $g(a) = f(\beta_1, ...\beta_{j-1}, a, \beta_{j+1}, ...\beta_p)$ we hope to find a^* s.t $h(a^*) = 0$ to minimize g(a) However the explicit solution of is hard to find.So we consider not to find the minimum of g(a),but to find a relative minimum of g(a) to make the object function $f(\beta)$ smaller after each iteration.We simply set$(\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j)(-x_{ij})^2)^{1/2}$ as constant in the update by giving $\beta_1^*, ...\beta_{i-1}^*, \beta_i^*, \beta_{i+1}^*, ...\beta_p^*$. We give the following update formula:

$$\beta_j^{k+1} \leftarrow S(\frac{\Sigma_{i=1}^n (y_i - \Sigma_{k\neq j} x_{ik}\beta_k^*) x_{ij}}{\Sigma_{i=1}^n x_{ij}^2}, \lambda \frac{(\Sigma_{i=1}^n (y_i - \Sigma_{k=1}^n x_{ik}\beta_k^*)^2)^{1/2}}{\Sigma_{i=1}^n x_{ij}^2})$$

where S(x,y) is a soft-threshold function and $f(\beta_1^{k+1}, ...\beta_{j-1}^{k+1}, \beta_j^k, \beta_{j+1}^k, ...\beta_p^k) = f(\beta_1^*, ...\beta_{i-1}^*, \beta_i^*, \beta_{i+1}^*, ...\beta_p^*)$

1.4 Numerical result and comparison

In this subsection,We use the linear regression model with standard normal errors $\epsilon_i \sim N(0, 1)$,set the true parameter value as $\beta_0 = (1, 1, 1, 1, 1, 0, 0, 0, ...)^t$,and vary $sigma$ between .25 and 3. p=500,n=100,generate regressors as $x_i \sim N(0, \Sigma)$ with the Toeplitz correlation matrix.The data is as same as the A.Belllon(2011)'s article. We use β_{SOCP} to represent the coef given by original method,and use β_{CD} the represent the coef given by our algorithm i

Fig.1 shows when we choose $\sigma = 1$,the convergence of our algorithm. $||\beta^k - \beta^{k-1}||_2$ with respect to the iteration number k.Fig.2 shows when σ varies the CPU time of 2 methods,Fig.3 show the difference between 2 results given by the 2 methods $||\beta_{SOCP} - \beta_{CD}||_2$

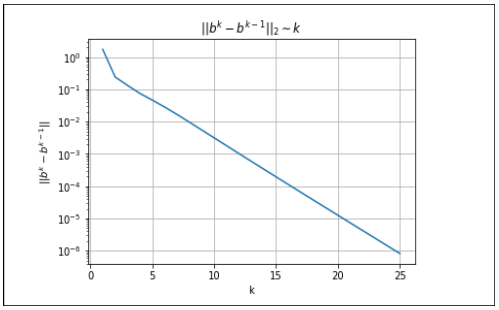

FIGURE 1: $||\beta^{k+1} - \beta^k||_2 \sim k$

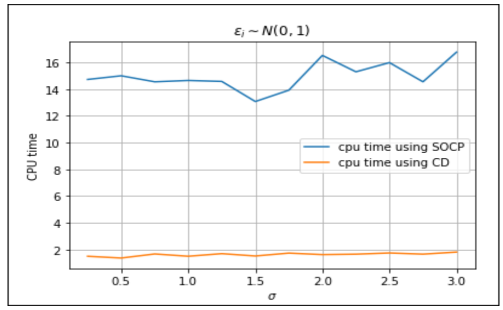

FIGURE 2: CPU time of SOCP methods

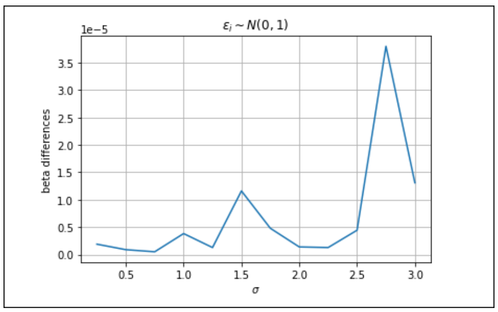

FIGURE 3: $||\beta_{SOCP} - \beta_{CD}||$

We see that our Algorithm converges very fast in fig1,$log(||\beta^k - \beta^{k-1}||_2) \sim k$ which means $||\beta^k - \beta^{k-1}||_2 \sim O(e^{-k})$.And after no more than 30 iterations,$||\beta^k - \beta^{k-1}||_2 \leq 1e-6$ From Fig2 we see that our algorithm run much faster than SOCP method,the average time on CPU consume by our algorithm $T_{CD} = 1.6225s$ and the traditional SOCP method need $T_{SOCP} = 14.9689s$,We make A.Bellon(2011)'s SOCP method on square-root lasso 9.2256 times faster..Fig3 shows that the coefficients generate by 2 algorithm is almost the same,which means our algorithm has a good accuracy.