Journal of Machine Learning Research 15 (2014) 489-493 Submitted 3/13; Revised 8/13; Published 2/14

The FASTCLIME Package for Linear Programming and Large-Scale Precision Matrix Estimation in R

Haotian Pang HPANG@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Olden St
Princeton, NJ 08540, USA

Han Liu HANLIU@PRINCETON.EDU
Robert Vanderbei RVDB@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
98 Charlton St
Princeton, NJ 08540, USA

Editor: Antti Honkela

Abstract

We develop an R package FASTCLIME for solving a family of regularized linear programming (LP) problems. Our package efficiently implements the parametric simplex algorithm, which provides a scalable and sophisticated tool for solving large-scale linear programs. As an illustrative example, one use of our LP solver is to implement an important sparse precision matrix estimation method called *CLIME* (Constrained L_1 Minimization Estimator). Compared with existing packages for this problem such as CLIME and FLARE, our package has three advantages: (1) it efficiently calculates the full piecewise-linear regularization path; (2) it provides an accurate dual certificate as stopping criterion; (3) it is completely coded in C and is highly portable. This package is designed to be useful to statisticians and machine learning researchers for solving a wide range of problems.

Keywords: high dimensional data, sparse precision matrix, linear programming, parametric simplex method, undirected graphical model

1. Introduction and Parametric Simplex Method

We introduce an R package, FASTCLIME, that efficiently solves a family of regularized LP problems. Our package has two major components. First, we provide an interface function that implements the parametric simplex method (PSM). This algorithm can efficiently solve large-scale LPs. Second, we apply the PSM to implement an important sparse precision matrix estimation method called CLIME (Cai et al., 2011), which is useful in high-dimensional graphical models. In the rest of this section, we describe briefly the main idea of the PSM. We refer readers who are unfamiliar with simplex methods in general to Vanderbei (2008). Consider the LP problem

$$\max c^T\beta \quad \text{subject to: } A\beta \le b,\ \ \beta \ge 0,$$

where $A \in \mathbb{R}^{n\times d}$, $c \in \mathbb{R}^d$ and $b \in \mathbb{R}^n$ are given and, "$\geq$" and "$\leq$" are defined componentwise. Simplex methods expect problems in "equality form". Therefore, the first task is to introduce new variables which we tack onto the end of β and make it a longer vector $\hat{\beta}$. We rewrite the constraints with the new variables on the left and the old ones on the right:

$$\max c^T\beta_N \quad \text{subject to: } \beta_B = b - A\beta_N,\ \beta_B \geq 0,\ \beta_N \geq 0.$$

Here, $N = \{1, 2, \ldots, d\}$, $B = \{d+1, d+2, \ldots, d+n\}$, and β_B and β_N denote subvectors of $\hat{\beta}$ associated with the indices in the set. In each iteration, one variable on the left is swapped with one on the right. In general, the variables on the left are called *basic variables* and the variables on the right are called *nonbasic variables*. As the algorithm progresses, the set of nonbasic variables changes and the objective function is re-expressed purely in terms of the current nonbasic variables and therefore the coefficients for the objective function change. In a similar manner, the coefficients in the linear equality constraints also change. We denote these changed quantities by $\hat{A}$, $\hat{b}$, and $\hat{c}$. Associated with each of these updated representations of the equations of the problem is a particular candidate "solution" obtained by setting $\beta_N = 0$ and reading off the corresponding values for basic variables $\beta_B = \hat{b}$. If $\hat{b} \geq 0$, then the candidate solution is feasible (that is, satisfies all constraints). If in addition $\hat{c} \leq 0$, then the solution is optimal.

Each variant of the simplex method is defined by the rule for choosing the pair of variables to swap at each iteration. The PSM's rule is described as follows (Vanderbei, 2008). Before the algorithm starts, it parametrically perturbs b and c:

$$\max(c + \lambda c^*)^T\beta \quad \text{subject to: } A\beta \leq b + \lambda b^*,\ \ \beta \geq 0. \tag{1}$$

Here $b^* \geq 0$ and $c^* \leq 0$; they are called *perturbation vectors*. With this choice, the perturbed problem is optimal for large λ. The method then uses pivots to systematically reduce λ to smaller values while maintaining optimality as it goes. Once the interval of optimal λ values covers zero, we simply set $\lambda = 0$ and read off the solution to the original problem. Sometimes there is a natural choice of the perturbation vectors b^* and c^* suggested by the underlying problem for which it is known that the initial solution to the perturbed problem is optimal for some value of λ. Otherwise, the solver generates perturbations on its own.

If we are only interested in solving generic LPs, the PSM is comparable to any other variant of the simplex method. However, as we will see in the next section, the parametric simplex method is particularly well-suited to machine learning problems since the relaxation parameter λ in (1) is naturally related to the regularization parameter in sparse learning problems. This connection allows us to solve a full range of learning problems corresponding to all the regularization parameters. If a regularized learning problem can be formulated as (1), then the entire solution path can be obtained by solving **one** LP with the PSM. More precisely, at each iteration of the PSM, the current "solution" is the optimal solution for some interval of λ values. If these solutions are stored, then when λ reaches 0, we have the optimal solution to every λ-perturbed problem for all λ between 0 and the starting value.

We describe the application of the PSM to sparse precision matrix estimation in Section 2. Numerical benchmark and comparisons with other implementations of the precision matrix estimation are provided in Section 3. For details of examples and how to use the package, we refer the user to the companion vignette and package references.

2. Application to Sparse Precision Matrix Estimation

Estimating large covariance and precision matrices is a fundamental problem which has many applications in modern statistics and machine learning. We denote $\Omega = \Sigma^{-1}$, where Σ is the population covariance matrix. Under Gaussian model, the sparse precision matrix Ω encodes the conditional independence relationships among the variables and that is why sparse precision matrices are closely related to undirected graphs. Recently, several sparse precision matrix estimation methods have been proposed, including penalized maximum-likelihood estimation (MLE) (Banerjee et al., 2008; Friedman et al., 2007b,a, 2010), neighborhood selection method (Meinshausen and Bühlmann, 2006) and LP based methods (Cai et al., 2011; Yuan, 2011). In general, solvers based on penalized MLE methods, such as `QUIC` (Hsieh et al., 2011) and `HUGE` (Zhao and Liu, 2012), are faster than the others. However, these MLE methods aim to find an approximate solution quickly whereas the linear programming methods are designed to find solutions that are correct essentially to machine precision. The comparison of classification performance can be found in Cai et al. (2011) and it is shown that CLIME uniformly outperforms the MLE methods. Because of the good theoretical properties shown by CLIME, we would like to develop a fast algorithm for implementing this method which serves as an important building block for more sophisticated learning algorithms.

The CLIME solves the following optimization problem

$$\min \|\Omega\|_1 \quad \text{subject to: } \|\Sigma_n\Omega - I_d\|_{\max} \le \lambda \ \text{ and } \Omega \in \mathbb{R}^{d\times d},$$

where I_d is the d-dimensional identity matrix, Σ_n is the sample covariance matrix, and $\lambda > 0$ is a tuning parameter. Here $\|\Omega\|_1 = \sum_{j,k} |\Omega|_{j,k}$ and $\|\cdot\|_{\max}$ is the elementwise sup-norm. This minimization problem can be further decomposed into d smaller problems, which allows us to recover the precision matrix in a column by column fashion. For the i-th subproblem, we get the i-th column of Ω, denoted as $\hat{\beta}$, by solving

$$\min \|\beta\|_1 \quad \text{subject to: } \|\Sigma_n\beta - e_i\|_\infty \le \lambda \ \text{ and } \beta \in \mathbb{R}^d, \tag{2}$$

where $\|\beta\|_1 = \sum_{j=1}^d |\beta_j|$ and $e_i \in \mathbb{R}^d$ is the i-th basis vector.

The original CLIME package manually sets a default path for λ and solves the LP problem for each different value of λ. In this paper, we propose to use the PSM to solve this problem more efficiently. CLIME can be easily formulated in parametric simplex LP form. Let β^+ and β^- be the positive and negative parts of β. Since $\beta = \beta^+ - \beta^-$ and $\|\beta\|_1 = \beta^+ + \beta^-$, Equation (2) becomes:

$$\min \beta_+ + \beta_- \quad \text{subject to: } \begin{pmatrix} \Sigma_n & -\Sigma_n \\ -\Sigma_n & \Sigma_n \end{pmatrix} \begin{pmatrix} \beta^+ \\ \beta^- \end{pmatrix} \le \begin{pmatrix} \lambda + e_i \\ \lambda - e_i \end{pmatrix}. \tag{3}$$

Comparing (1) and (3), we can give the following identification:

$$A = \begin{pmatrix} \Sigma_n & -\Sigma_n \\ -\Sigma_n & \Sigma_n \end{pmatrix}, \quad b = \begin{pmatrix} e_i \\ -e_i \end{pmatrix}, \quad c = \begin{pmatrix} -1 \\ \vdots \\ -1 \end{pmatrix}, \quad b^* = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \quad c^* = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The path of λ defined by the PSM corresponds to the path of λ as described in CLIME. Therefore, CLIME can be solved efficiently by the PSM; furthermore, when the optimal solution is sparse, the parametric simplex is able to find the optimal solution after very few iterations.

3. Performance Benchmark

For our experiments we focused solely on CLIME. We compare the timing performance of our package with the packages flare and clime. Flare uses the Alternating Direction Method of Multipliers (ADMM) algorithm to evaluate CLIME (Li et al., 2012), whereas clime solves a sequence of LP problems for a certain specific set of values of λ. As explained in Section 1, our method calculates the solution for *all* λ, while flare and clime use a discrete set of λ values as specified in the function. We fix the sample size n to be 200 and vary the data dimension d from 50 to 800. We generate our data using `fastclime.generator`, without any particular data structures. Clime and fastclime are based on algorithms that solve problems to machine precision (10^{-5}). Flare, on the other hand, is an ADMM-based algorithm that stops when the change from one iteration to the next drops below the same threshold. As shown in Table 1, fastclime performances significantly faster than clime when d equals 50 and 100. When d becomes large, we are not able to obtain results from clime in one hour. We also notice that, in most cases, fastclime performances consistently better than flare, and it has a smaller deviation compared with flare. The reason fastclime outperforms the other methods is primarily because the PSM only solves one LP problem to get the entire solution path for all λ quickly and without using much memory. The code is implemented on a i5-3320 2.6GHz computer with 8G RAM, and the R version used is 2.15.0.

Method	d=50	d=100	d=200	d=400	d=800
clime	103.52(9.11)	937.37(6.77)	N/A	N/A	N/A
flare	0.632(0.335)	1.886(0.755)	10.770(0.184)	74.106(33.940)	763.632(135.724)
fastclime	0.248(0.0148)	0.928(0.0268)	9.928(3.702)	53.038(1.488)	386.880(58.210)

Table 1: Average Timing Performance of Three Solvers in Seconds

4. Summary and Acknowledgements

We developed a new package named fastclime, for solving linear programming problems with a relaxation parameter and high dimensional sparse precision matrix estimation. We plan to maintain and support this package in the future. Han Liu is supported by NSF Grants III-1116730 and NSF III-1332109, NIH R01MH102339, NIH R01GM083084, and NIH R01HG06841, and FDA HHSF223201000072C. Robert Vanderbei is supported by ONR Grant N000141310093.

References

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 9:485–516, 2008.

T. Cai, W. Liu, and X. Luo. A constrained l_1 minimization approach to sparse precision matrix estimation. *J. American Statistical Association*, 106:594–607, 2011.

J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007a.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2007b.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

C-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. *Advances in Neural Information Processing Systems*, 24, 2011.

X. Li, T. Zhao, X. Yuan, and H. Liu. An R package flare for high dimensional linear regression and precision matrix estimator. *R Package Vigette*, 2012.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

R. Vanderbei. *Linear Programming, Fundations and Extensions*. Springer, 2008.

M. Yuan. High dimensional inverse covariance matrix estimation via linear programming. *Journal of Machine Learning Research*, 11:2261–2286, 2011.

T. Zhao and H. Liu. The huge package for high-dimensional undirected graph estimation in R. *Journal of Machine Learning Research*, 13:1059–1062, 2012.