Robust Parameter Estimation for Regularized Structured Equation Modeling

Yilun Zhang and Xiaodong Cai

University of Miami, Miami, FL

ARTICLE HISTORY

Compiled January 3, 2019

ABSTRACT

Regularized structural equation modeling combines the statistical inference of structural equation modeling with regression shrinkage technique.The objective function is to minimize a penalized negative log-likelihood function. The penalty terms like Lasso and SCAD can shrink some of the estimated parameters to zero and induce a sparse estimation. The difficulty is that objective function is both non-convex and non-smooth which challenges the convergence of many optimization methods. We propose to use gradient sampling to solve regularized SEM. A provable convergence can be obtained for a broad family of penalty terms. We compare gradient sampling with the performances of other optimization methods in computer simulation and real data experiments. The result shows gradient sampling has a competitive performance on parameter estimation of regularized SEM. An R package SEMGS implementing the proposed method is freely available on the Internet.

KEYWORDS

regularized SEM, Lasso, gradient sampling , non-smooth non-convex optimization, convergence

1. Introduction

Structural equation modeling (SEM) is a statistical approach for modeling multivariate structural relationships (Bollen, 1989; Kaplan, 2000; Lee, 2007). It is a melding of factor analysis and simultaneous equation modeling with latent variables into one comprehensive statistical methodology. The conventional approach to SEM as generally practiced in social and behavior sciences employs confirmatory factor analysis (CFA) and is apparently confirmatory in nature. In this approach, an available theory is represented by a structural equation model, model parameters are estimated with

Correspondence should be addressed to Xiaodong Cai, Department of Electrical and Computer Engineering, University of Miami, FL 33146, USA. Email: x.cai@miami. edu

the observed data, and then an assessment of the goodness-of-fit of the model is performed followed by model modification if necessary (Kaplan, 2000). While the CFA modeling that incorporates *a prior* substantive knowledge makes the definition of the latent variables better grounded in subject-matter theory and leads to parsimonious models, this approach often forces a researcher to specify a model too parsimonious to fit the data well. To overcome this problem, the exploratory SEM (ESEM) was developed recently to extend SEM to allow less restrictive exploratory factor analysis (EFA) models to be used in conjunction with the traditional CFA models (Asparouhov & Muthén, 2009; Marsh, Morin, Parker, & Kaur, 2014). While ESEM enables more flexible modeling, it introduces more free parameters to be estimated. If a traditional estimation method such as the maximum likelihood (ML) method is employed, the large number of free parameters can reduce the efficiency of the estimator, and lead to a dense loading matrix that may be less interpretable than the sparse loading matrix in traditional SEM.

Introduction of exploratory factors in ESEM essentially increases the dimension of the model space. To mitigate the problem of traditional estimation methods mentioned earlier, we can use the supervised learning approach in machine learning (Bishop, 2006; Hastie et al., 2009) to find a parsimonious model from the model space, that offers the best trade-off between the model complexity and the estimation error. The supervised learning approach typically employs cross validation to determine a model that yields the smallest prediction error, or uses an information criterion such as the Bayesian information criterion to trade complexity for the estimation error. One technique often used to control the model complexity is regularization or shrinkage (Bishop, 2006; Hastie et al., 2009), which attempts to reduce the values of model parameters by adding a penalty term to the error function that is to be minimized to estimate model parameters. In particular, the ℓ_1-regularization, also named as lasso in the context of linear regression (Hastie, Tibshirani, & Wainwright, 2015; Tibshirani, 1996), can shrink values of parameters to zero, yielding a sparse or parsimonious model.

Recently, ℓ_1-regularized estimation methods have been applied to SEM (Huang, Chen, & Weng, 2017; Jacobucci, 2017). Computer simulations and an empirical study (Jacobucci, 2017) demonstrated that the ℓ_1-regularized method, named RegSEM, allows for modeling flexibility to find the model structure that increases both interpretability and generalizability. A penalized ML method was developed for SEM estimation (Huang et al., 2017), and numerical experiments showed that penalized ML method outperforms the traditional ML method in the estimation of ESEM models. These studies clearly demonstrate the utility of the ℓ_1-regularized SEM. Of note, the regularized SEM not only is useful in model selection as shown in (Huang et al., 2017; Jacobucci, 2017), but also may be able to handle the problem of "small N and large p", where the number of model parameter p is much larger than the number of observations N. In linear regression, when the model is sparse, or more specifically, when

$p_0 (p_0 < N \ll p)$ of p regression coefficients are nonzero, lasso is consistent in selecting these p_0 variables (Hastie et al., 2015), if the design matrix satisfies certain condition. Although it is unclear if the regularized SEM can be consistent in model estimation, it may yield a good estimate of the model if the SEM model is sparse, whereas the traditional ML method breaks down when $N < p$.

The regularized SEM approaches in (Huang et al., 2017; Jacobucci, 2017) estimate model parameters by maximizing the sum of the likelihood function and a regularization term, which typically is the ℓ_1-norm of the exploratory variables, although other regularizers such as SCAD (Fan & Li, 2001), elastic net (Zou & Hastie, 2005), or adaptive lasso (Zou, 2006) can also be used. Since the likelihood is nonconvex and the regularizer is nonsmooth, the objective function is apparently nonconvex and nonsmooth. The RegSem (Jacobucci, 2017) uses the proximal gradient method or the coordinate-wise proximal method (Parikh, Boyd, et al., 2014), as implemented in the *regsem* software package (Jacobucci, Grimm, Brandmaier, & Serang, n.d.), to solve the optimization method. The penalized ML method (Huang et al., 2017), implemented in the *lsl* software package (Huang, n.d.), uses the expectation-maximization (EM) algorithm, where the maximization step is implemented with the coordinate-wise proximal method. If the objective function satisfies certain restrictive conditions (Attouch, Bolte, Redont, & Soubeyran, 2010; Attouch, Bolte, & Svaiter, 2013; Bolte, Sabach, & Teboulle, 2014; Ochs, Chen, Brox, & Pock, 2014; Parikh et al., 2014; Tao et al., 2005; Xu & Yin, 2013), the proximal gradient method or the coordinate-wise proximal method can converge. However, these conditions generally do not hold for regularized SEM, or it is difficult to verify if a SEM model to be estimated meets these conditions. In fact, we observed in our computer simulations that the algorithms in software packages *regsem* and *lsl* often do not converge, yielding unacceptable models. Therefore, more robust estimation methods are needed for the regularized SEM.

In this paper, we apply the gradient sampling (GS) approach (J. R. Burke, Lewis, & Overton, 2005; Kiwiel, 2007) to minimize the nonconvex and nonsmooth objective function for the estimation of regularized SEM models. The GS approach is guaranteed to converge to a stationary point of the objective function with probability one under mild conditions. It can be verified that the objective function of the regularized SEM satisfies the convergence condition for commonly used regularizers. Therefore, the GS approach can yield robust estimation of regularized SEM models.

The remaining part of the paper is organized as follows. In Section 2, the regularized SEM is introduced and associated optimization methods for model estimation are described briefly. In Section 3, the gradient sampling method for SEM estimation is developed. In Section 4, computer simulations and real data analysis are carried out to compare the performance of our GS method with that of regsem and lsl. Finally, conclusions are drawn in Section 5.

2. Regularized SEM

We consider the general structural equation model with continuous latent variables, which consists of two components: a structural equation specifying the relations among latent variables, and measurement equations linking the latent variable to the observable variables (Bollen, 1989; Kaplan, 2000; Lee, 2007). The structural equation is given by

$$\boldsymbol{\eta} = \boldsymbol{\Pi}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\epsilon}_0, \tag{1}$$

where $\boldsymbol{\eta} \in \mathbb{R}^{r\times 1}$ is a vector of endogenous latent variables, $\boldsymbol{\xi} \in \mathbb{R}^{s\times 1}$ is a vector of exogenous latent variables, $\boldsymbol{\Pi}$ is a $r \times r$ matrix of regression coefficients, $\boldsymbol{\Gamma}$ is a $r \times s$ matrix of regression coefficients, and $\boldsymbol{\epsilon}_0$ is the residual term. It is assumed that diagonal elements of $\boldsymbol{\Pi}$ are zero and $\mathbf{I} - \boldsymbol{\Pi}$ is nonsigular. The measurement equations are given by

$$\begin{aligned} \mathbf{x}_1 &= \boldsymbol{\Lambda}_1\boldsymbol{\eta} + \boldsymbol{\epsilon}_1, \\ \mathbf{x}_2 &= \boldsymbol{\Lambda}_2\boldsymbol{\xi} + \boldsymbol{\epsilon}_2, \end{aligned} \tag{2}$$

$\mathbf{x}_1 \in \mathbb{R}^{p\times 1}$ and $\mathbf{x}_2 \in \mathbb{R}^{q\times 1}$ are vectors of manifest variables for $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$, respectively, $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$ are loading matrices, $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are vectors of random measurement errors. It is assumed that $\boldsymbol{\epsilon}_0$, $\boldsymbol{\epsilon}_1$, and $\boldsymbol{\epsilon}_2$ are independent random vectors with zero mean and covariance $\boldsymbol{\Psi}_0$, $\boldsymbol{\Psi}_1$, and $\boldsymbol{\Psi}_2$, respectively; $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are assumed to be diagonal (Bollen, 1989), while $\boldsymbol{\Psi}_0$ is not necessarily diagonal. It is also assumed that $\boldsymbol{\xi}$ is Gaussian with zero mean and covariance matrix $\boldsymbol{\Phi}$ and is uncorrelated with $\boldsymbol{\epsilon}_0$, $\boldsymbol{\epsilon}_1$, and $\boldsymbol{\epsilon}_2$.

In traditional SEM, the measurement equations (2) follow the CFA model. Here, we allow for more flexible ESEM by including some exploratory factors (Asparouhov & Muthén, 2009). The sub-matrices of $\boldsymbol{\Lambda}_1$, $\boldsymbol{\Lambda}_2$, $\boldsymbol{\Pi}$, and $\boldsymbol{\Gamma}$ corresponding to the confirmatory factors are generally known to be sparse, since they are specified based on a prior substantive knowledge. On the other hand, The sub-matrices of $\boldsymbol{\Lambda}_1$, $\boldsymbol{\Lambda}_2$, $\boldsymbol{\Pi}$, and $\boldsymbol{\Gamma}$ corresponding to the exploratory variables generally contain free parameters, although partial knowledge can also be incorporated to restrict certain parameters. The goal of the regularized SEM is to find a parsimonious model, or equivalently sparse matrices $\boldsymbol{\Lambda}_1$, $\boldsymbol{\Lambda}_2$, $\boldsymbol{\Pi}$, and $\boldsymbol{\Gamma}$, that fits the data well.

Let $\boldsymbol{\theta}$ be the vector that contains all unknown model parameters in $\boldsymbol{\Lambda}_1$, $\boldsymbol{\Lambda}_2$, $\boldsymbol{\Pi}$, $\boldsymbol{\Gamma}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}_0$, $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$. In other words, those parameters in these matrices, that are fixed for model identification or other special purposes (Bollen, 1989; Lee, 2007), are not included in $\boldsymbol{\theta}$. Matrix $\boldsymbol{\Psi}_0$ is constrained to be symmetric. Define $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T]^T$, then the covariance matrix of $\mathbf{x}$, denoted as $\boldsymbol{\Sigma}(\boldsymbol{\theta})$, can be found from (1) and (2) as follows

(Bollen, 1989; Lee, 2007)

$$\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\Lambda}_1(\mathbf{I}-\boldsymbol{\Pi})^{-1}(\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Gamma}^T+\boldsymbol{\Psi}_0)((\mathbf{I}-\boldsymbol{\Pi})^{-1})^T\boldsymbol{\Lambda}_1^T+\boldsymbol{\Psi}_1 & \boldsymbol{\Lambda}_1(\mathbf{I}-\boldsymbol{\Pi})^{-1}\boldsymbol{\Gamma}\boldsymbol{\Phi}\boldsymbol{\Lambda}_2^T \\ \boldsymbol{\Lambda}_2\boldsymbol{\Phi}\boldsymbol{\Gamma}^T(\mathbf{I}-\boldsymbol{\Pi})^{-1})^T\boldsymbol{\Lambda}_1^T & \boldsymbol{\Lambda}_2\boldsymbol{\Phi}\boldsymbol{\Lambda}_2^T+\boldsymbol{\Psi}_2, \end{bmatrix}. \tag{3}$$

Suppose that N observations $\mathbf{x}_i, i = 1, \ldots, N$, are available. The sample covariance $\mathbf{S}$ is given by $\mathbf{S} = \frac{1}{N}\sum_{i=1}^N(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$, where $\bar{\mathbf{x}} = \sum_{i=1}^N \mathbf{x}_i/N$. The ML method estimates parameter $\boldsymbol{\theta}$ by minimizing the negative log-likelihood function (Bollen, 1989):

$$f(\boldsymbol{\theta}) = \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \mathrm{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta})), \tag{4}$$

where $\mathrm{tr}(\cdot)$ represent the matrix trace.

The regularized SEM (Jacobucci, Grimm, & McArdle, 2016) adds a regularization term to $f(\boldsymbol{\theta})$ and minimizes the following objective function

$$F(\boldsymbol{\theta}) = \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \mathrm{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta})) + \lambda P(\boldsymbol{\theta}_s), \tag{5}$$

where $\boldsymbol{\theta}_s$ is a vector containing a subset s of elements of $\boldsymbol{\theta}$, $\lambda > 0$ is a constant, and the regularization function $\lambda P(\boldsymbol{\theta}_s)$ will be defined in the following. Typically, $\boldsymbol{\theta}_s$ contains parameters in $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$, $\boldsymbol{\Pi}$, and $\boldsymbol{\Gamma}$ that are associated with exploratory variables, although it is possible to include other parameters in $\boldsymbol{\theta}_s$. The purpose of introducing $\lambda P(\boldsymbol{\theta}_s)$ is to penalize the parameters in $\boldsymbol{\theta}_s$, aiming to yield a sparse model. Several well-known penalty functions in linear regression can be adopted here. The lasso penalty is given by (Tibshirani, 1996)

$$P(\boldsymbol{\theta}_s) = \|\boldsymbol{\theta}_s\|_1, \tag{6}$$

where $\|\cdot\|_1$ stands for the ℓ_1-norm. The elastic net penalty is (Zou & Hastie, 2005)

$$P_\alpha(\boldsymbol{\theta}_s) = \alpha\|\boldsymbol{\theta}_s\|_1 + \frac{1-\alpha}{2}\|\boldsymbol{\theta}_s\|_2^2, \tag{7}$$

where $0 \le \alpha \le 1$ and $\|\cdot\|_2$ stands for the Euclidean norm. The SCAD penalty (Fan & Li, 2001) can also be used. Note that all these penalty functions are not smooth. Since $f(\boldsymbol{\theta})$ is not convex, $F(\boldsymbol{\theta})$ is nonconvex and nonsmooth.

To estimate the model parameters for RegSem, the software *regsem* (Jacobucci et al., n.d.) implements several unconstrained optimization algorithms to minimize $F(\boldsymbol{\theta})$. These algorithms include the proximal gradient (**pg**) method (Parikh et al., 2014), the proximal coordinate descent (**pcd**) method (Tseng & Yun, 2009) which is also named as proximal alternating linearized minimization (PALM) (Bolte et al.,

2014), and two variants of the pcd method, named the proximal coordinate descent with line search (**pcdl**) (Beck & Teboulle, 2009; Parikh et al., 2014) and proximal coordinate descent with momentum (**pcdm**) (Beck & Teboulle, 2009; Boţ, Csetnek, & László, 2016). While these algorithms converge for convex optimization problems, it is not clear if they converge to a stationary point for the nonconvex function $F(\boldsymbol{\theta})$. Moreover, covariance matrices $\boldsymbol{\Psi}_0$, $\boldsymbol{\Psi}_1$, $\boldsymbol{\Psi}_2$, and $\boldsymbol{\Phi}$ are positive definite, and $\mathbf{I}-\boldsymbol{\Pi}$ is nonsingular. However, such constraints are not taken into account by these algorithms. Therefore, it is possible that algorithms may not converge or converge to a solution that violates the constraints. In our computer simulations, we observed that these algorithm often did not converges. In the next section, we will apply the robust GS method (J. R. Burke et al., 2005; Kiwiel, 2007) to minimize $F(\boldsymbol{\theta})$, which can converge to a stationary point of $F(\boldsymbol{\theta})$ with probability one.

3. The Gradient Sampling Method for Regularized SEM

Function $F(\boldsymbol{\theta})$ in (5) is nonconvex and nonsmooth. It is well known that the steepest descent algorithm often converge to a non-stationary point when applied to nonsmooth objective functions, whether convex or not (Asl & Overton, 2017; Wolfe, 1975). The fundamental difficulty is that such nonsmooth objective functions usually have minimizers where the gradient is not defined, which is true for $F(\boldsymbol{\theta})$, because it is expected that the minimizer contains many zero elements where $F(\boldsymbol{\theta})$ is not differentiable. However, $F(\boldsymbol{\theta})$ is not differentiable at a finite number of points in its domain. In other words, $F(\boldsymbol{\theta})$ is differentiable almost everywhere, although it may be not differentiable at its minimizer. The GS method has been developed to handle this kind of objective function (J. R. Burke et al., 2005; Kiwiel, 2007) with proven convergence property. The idea of the GS method is that at a given iterate, we compute the gradient of the objective function on a set of randomly generated nearby points, and use these gradients to construct a descent direction. Next, we will formulate our optimization problem taking into account the positive definite constraint on $\boldsymbol{\Psi}_0$, $\boldsymbol{\Psi}_1$, $\boldsymbol{\Psi}_2$, and $\boldsymbol{\Phi}$, and the nonsingular constraint on $\mathbf{I}-\boldsymbol{\Pi}$, and then apply the GS method to solve the optimization problem.

Matrices $\boldsymbol{\Psi}_1$ and $\boldsymbol{\Psi}_2$ are diagonal and their diagonal entries should be positive. Let ψ_{1i}, $i=1,\ldots,p$ be the diagonal entries of $\boldsymbol{\Psi}_1$, and ψ_{2i}, $i=1,\ldots,q$ be the diagonal entries of $\boldsymbol{\Psi}_2$. Then, we have $\psi_{1i}>0$, $i=1,\ldots,p$, and $\psi_{2i}>0$, $i=1,\ldots,q$. Since $\boldsymbol{\Psi}_0$ and $\boldsymbol{\Phi}$ are positive definite, we can use Cholesky decompostion to write them as $\boldsymbol{\Psi}_0=\mathbf{L}\mathbf{L}^T$ and $\boldsymbol{\Phi}=\mathbf{P}\mathbf{P}^T$, where both $\mathbf{L}$ and $\mathbf{P}$ are lower triangle matrices with positive diagonal entries, i.e., $L_{ii}>0, i=1,\ldots,r$, and $P_{ii}>0, i=1,\ldots,s$. We will relax the nonsingular constraint on $\mathbf{I}-\boldsymbol{\Pi}$ as $|\mathbf{I}-\boldsymbol{\Pi}|^2>\gamma$, where γ is a small positive constant. With these definitions, $\boldsymbol{\Psi}_0$ and $\boldsymbol{\Phi}$ in the parameter vector $\boldsymbol{\theta}$ are replaced by

$\mathbf{L}$ and $\mathbf{P}$. Then the optimization problem for regularized SEM is

$$\begin{array}{ll} \underset{\boldsymbol{\theta}\in\mathbb{R}^n}{\text{minimize}} & F(\boldsymbol{\theta}) = \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})| + \text{tr}(\mathbf{S}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\theta})) + \lambda P(\boldsymbol{\theta}_s), \\ \text{subject to} & -\psi_{1i} < 0, i = 1, \ldots, p \\ & -\psi_{2i} < 0, i = 1, \ldots, q \\ & -L_{ii} < 0, i = 1, \ldots, r \\ & -P_{ii} < 0, i = 1, \ldots, s \\ & -|\mathbf{I} - \boldsymbol{\Pi}|^2 + \gamma \leq 0, \end{array} \tag{8}$$

where n is the total number of free parameters to be estimated.

The original GS algorithm was developed to solve unconstrained optimization problems where the objective function can be nonsmooth and nonconvex (J. R. Burke et al., 2005), and several modifications were proposed to improve the GS algorithm for unconstrained optimization (Kiwiel, 2007). More recently, the GS method was combined with sequential quadratic programming (SQP) for solving constrained optimization problems (Curtis & Overton, 2012; Tang, Liu, Jian, & Li, 2014). We will apply the feasible SQP-GS (FSQP-GS) method (Tang et al., 2014) to solve the optimization (8). At an iterate $\boldsymbol{\theta}^k$, FSQP-GS executes three steps: 1) random sampling to obtain $2n$ points in the neighborhood of $\boldsymbol{\theta}^k$, 2) solve a quadratic programming problem to find a feasible descent direction, and 3) line search to determine the step size.

Before describing these three steps, let us first find the derivative of $F(\boldsymbol{\theta})$. Let us assume that the lasso penalty is used in $F(\boldsymbol{\theta})$, i.e., $P(\boldsymbol{\theta}_s) = \|\boldsymbol{\theta}_s\|_1$. Recall that s denotes the set of regularized parameters, then the derivative of $F(\boldsymbol{\theta})$ on its differentiable domain is given by

$$\frac{\partial F(\boldsymbol{\theta})}{\partial \theta_i} = \begin{cases} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} + \lambda \,\text{sign}(\theta_i), & \text{if } \ i \in s \\ \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i}, & \text{otherwise,} \end{cases} \tag{9}$$

where $\text{sign}(\theta_i) = 1$ if $\theta_i > 0$, or -1 if $\theta_i < 0$. The partial derivatives of $f(\boldsymbol{\theta})$ can be written as

$$\begin{aligned} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_i} &= \frac{\partial \log|\boldsymbol{\Sigma}(\boldsymbol{\theta})|}{\partial \theta_i} + \frac{\partial \text{tr}(\mathbf{S}\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1})}{\partial \theta_i} \\ &= \text{tr}\Big(\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_i}\Big) - \text{tr}\Big(\mathbf{S}\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\theta})}{\partial \theta_i}\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\Big). \end{aligned} \tag{10}$$

Note that it is not difficult to derive $\partial\boldsymbol{\Sigma}(\boldsymbol{\theta})/\partial\theta_i$ from $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ given in (3). It is seen from (10) that if both $\mathbf{I} - \boldsymbol{\Pi}$ and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ are invertible, then $f(\boldsymbol{\theta})$ is continuously differentiable and its derivative is locally bounded, and therefore it is locally Lipschitz. Let $\mathcal{D}^F$ denote the open dense subset of $\mathbb{R}^d$ over which $F(\boldsymbol{\theta})$ is locally Lipschitz and continuously differentiable. We can write the constraints in (8) as $c_i(\boldsymbol{\theta}) < 0$, $i = 1, \cdots, n_c$,

where n_c is the number of constraints. Apparently, every function $c_i(\boldsymbol{\theta})$ is continuously differentiable and locally Lipschitz. Define $\mathbb{B}_\epsilon(\boldsymbol{\theta}^k) := \{\boldsymbol{\theta} \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}^k\|_2 < \epsilon\}$. If $\mathcal{D}^{c_i}$ is the open dense set over which $c_i(\boldsymbol{\theta})$ is continuously differentiable and locally Lipschitz, we apparently have $\mathbb{B}_\epsilon(\boldsymbol{\theta}^k) \cap \mathcal{D}^{c_i} = \mathbb{B}_\epsilon(\boldsymbol{\theta}^k)$. Now, we are ready to describe the three steps in each iteration of the FSQP-GS algorithm.

In the random sampling step, we randomly generate a set of n points in $\mathbb{B}_\epsilon(\boldsymbol{\theta}^k) \cap \mathcal{D}^F$, which is denoted as $\mathcal{B}_F^k$, and another set of n points in $\mathbb{B}_\epsilon(\boldsymbol{\theta}^k) \cap \mathcal{D}^{c_i} = \mathbb{B}_\epsilon(\boldsymbol{\theta}^k)$, which is denoted as $\mathcal{B}_c^k$. To describe the quadratic programming step, we first define $\psi_{1,\min}^k := \min\{\psi_{1i}^k, i = 1, \ldots, p\}$, $\psi_{2,\min}^k := \min\{\psi_{2i}^k, i = 1, \ldots, q\}$, $L_{ii,\min}^k := \min\{L_{ii}^k, i = 1, \ldots, r\}$, $P_{ii,\min}^k := \min\{P_{ii}^k, i = 1, \ldots, s\}$, and $c_{\min}^k := \min\{\psi_{1,\min}^k, \psi_{2,\min}^k, L_{ii,\min}^k, P_{ii,\min}^k\}$. If we define the function $c(\boldsymbol{\Pi}) := -|\mathbf{I} - \boldsymbol{\Pi}|^2 + \gamma$, then we have

$$\frac{\partial c(\boldsymbol{\Pi})}{\partial \boldsymbol{\Pi}} = 2|\mathbf{I} - \boldsymbol{\Pi}|\mathbf{C}_{\mathbf{I}-\boldsymbol{\Pi}},$$

where $\mathbf{C}_{\mathbf{I}-\boldsymbol{\Pi}}$ the is cofactor matrix of $\mathbf{I}-\boldsymbol{\Pi}$. Let $\mathbf{v}(\boldsymbol{\Pi})$ be the vector formed by stacking columns of $2|\mathbf{I} - \boldsymbol{\Pi}|\mathbf{C}_{\mathbf{I}-\boldsymbol{\Pi}}$ sequentially. Then, in the quadratic programming step, we solve the following optimization problem:

$$\begin{aligned} \underset{\mathbf{d}\in\mathbb{R}^n, z\in\mathbb{R}}{\text{minimize}} \quad & z + \frac{1}{2}\mathbf{d}^T\mathbf{H}^k\mathbf{d} \\ \text{subject to} \quad & \nabla F(\boldsymbol{\theta})^T\mathbf{d} \le z, \ \forall \boldsymbol{\theta} \in \mathcal{B}_F^k \\ & -c_{\min}^k < z \\ & c(\boldsymbol{\Pi}^k) + \mathbf{v}(\boldsymbol{\Pi})^T\mathbf{d}_{\boldsymbol{\Pi}} \le z, \ \forall \boldsymbol{\Pi} \in \mathcal{B}_c^k, \end{aligned} \tag{11}$$

where $\mathbf{H}^k$ is a symmetric, sufficiently positive definite, and bounded matrix, and $\mathbf{d}_{\boldsymbol{\Pi}}$ is the sub-vector of $\mathbf{d}$ that corresponds to $\boldsymbol{\Pi}$.

After the descend direction $\mathbf{d}$ is obtained, the third line search step is executed to determine the step-size α through the standard ArmijoWolfe procedure, which described in Procedure 3 (J. R. Burke et al., 2005). Typical values of β and η in Procedure 3 are $\beta = 0.5$ and $\eta = 0.01$. Finally, the gradient sampling method is summarized in Algorithm .

In each iteration of the gradient sampling algorithm, the major computation is to compute the gradient of the objective function at the sampled points and to solve the quadratic programing problem. Comparing with the gradient or proximal gradient method, where the gradient is evaluated only once per iteration, the gradient sampling method can significantly increase computation. However, we can reduce the computation of the gradient sampling method as follows. In iteration k, only the gradients at the points in $(B(\boldsymbol{\theta}_k, r) - B(\boldsymbol{\theta}_{k-1}, r)) \cap D$ need to be computed, and gradients at the points in $(B(\boldsymbol{\theta}_k, r) \cap B(\boldsymbol{\theta}_{k-1}, r)) \cap D$ can be reused. If we reuse these gradients,

Procedure 1 ArmijoWolfe line search:LS (g, $\boldsymbol{\theta}$, β):

Set $e \leftarrow -g/||g||_2$
for i=0, 1, 2, . . . **do**
  set $\alpha \leftarrow \beta^i$
  if $F(x + \alpha e) < F(x) - \eta\alpha||g||_2$ **then**
    if $(x + \alpha e) \in D$ **then** Stop
    else perturb x such that $(x + \alpha e) \in D \cap B(x_k, r)$
    end if
  end if
end for
return (α, e)

Algorithm Gradient Sampling

Pick $s \geq d + 1$, x^0 as the the initial value, r the sampling radius, $\beta \in (0, 1)$ the Armijo parameter, $\eta > 0$ the Wolfe parameter, , and ϵ the convergence threshold , $k \leftarrow 0$
while convergence not reached **do**
  $g^k \leftarrow$ DS(s, x^k, r), $(\alpha^k, e^k) \leftarrow$ LS(g^k, x^k, β)
  if $||g^k||_2 < \epsilon$ or $\langle \nabla F(x^k), g^k \rangle \leq 0$ **then** convergence reached
  else $x^{k+1} \leftarrow x^k - \alpha^k e^k$
  end if
end while

the computation burden of evaluating gradients can be reduced significantly. For the quadratic programming problem, we can use the solution in the previous iteration as a warm start for the current iteration. Because the previous solution may be close to the optimal solution of the quadratic programing in the current iteration, using it as the initial values can reduce the number of iterations (J. Burke, Lewis, & Overton, 2006; Curtis & Que, 2013).

Another issue is that the gradient sampling algorithm in Algorithm generally cannot shrink variables exactly to zero, although using the regularization such as lasso is expected to yield a sparse solution. We next describe how to add a simple step to the end of Algorithm to obtain a sparse solution. To illustrate how gradient sampling works under sparse induced penalty, especially when the objective function is not differentiable at the optimal point where some of the coefficients take zeros, we consider an l_1 norm minimization problem, and show how the descent direction obtained by gradient sampling can shrink the coordinate to take zero. The objective function is $F(x_1, x_2) = |x_1| + |x_2|$ where f is smooth. The set of differentiable points D is $\{(x_1, x_2) \in R^2 | x_1 \neq 0, \text{and } x_2 \neq 0\}$. And $s \geq 3$ points are sampled in the iteration.

Suppose we reach (a, b) where $|a| > r$ and $|b| > r$ and sampling radius is $r << 1$. Then $\frac{\partial F}{\partial x1} = 1$ and $\frac{\partial F}{\partial x2} = 1$. The direction for descent is $(1, 1)$ and it is the same as the gradient.

Now suppose we reach (a, b) where $|a| > r$ and $0 \leq b < r$this is the case where the

sampling ball contains non-differentiable points. The gradient at (a, b) is still $(1, 1)$. However, the direction given by gradient sampling can have three cases. The set of sampled gradients can be $(1, 1)$, $(1, 1), (1, -1)$, or $(1, -1)$ some the second coordinate of the sampled points are negative. In the case where the sampled points make the set $(1, 1), (1, -1)$, whose convex hull is then the line segment between these two points, the descent direction is $(1, 0)$. See Figure 1 for an illustration. We can compute that, the probability that the descent direction is $(1, 1), (1, 0), (1, -1)$ is $(\frac{1}{2} + \frac{b}{r})^s, 1 - (\frac{1}{2} + \frac{b}{r})^s - (\frac{1}{2} - \frac{b}{r})^s, (\frac{1}{2} - \frac{b}{r})^s$ respectively. So the larger the b, the larger the probability that x_2 will be shrunk. And when x_2 is closer to 0, it will have more probability to be fixed since the derivative on the second coordinate is zero. So eventually after many iterations, x_2 will reach zero, within an error that can be ignored numerically. Similar analysis can be done for the first coordinate x_1, and finally the gradient sampling will iterate to the point $(0, 0)$, which is the optimal point.

4. Experiment

In this section we conduct several simulation experiments to compare the performance of the proposed method with other methods, and apply it to a gene network inference problem.

4.1. Simulation study

Recall in the section 1, r is the number of endogenous variables and s is the number of exogenous variables. For each of the $r+s$ latent variables, it is measured by m number of manifest variables. For simulation I, the model parameters are set as followed

$$\mathbf{\Lambda}_1 = \begin{bmatrix} \textcircled{1} & * & * \\ \theta^1_{2,1} & * & * \\ \theta^1_{3,1} & * & * \\ * & \textcircled{1} & * \\ * & \theta^1_{5,2} & * \\ * & \theta^1_{6,2} & * \\ * & * & \textcircled{1} \\ * & * & \theta^1_{8,3} \\ * & * & \theta^1_{9,3} \end{bmatrix}, \mathbf{\Lambda}_2 = \begin{bmatrix} \textcircled{1} & * & * \\ \theta^2_{2,1} & * & * \\ \theta^2_{3,1} & * & * \\ * & \textcircled{1} & * \\ * & \theta^2_{5,2} & * \\ * & \theta^2_{6,2} & * \\ * & * & \textcircled{1} \\ * & * & \theta^2_{8,3} \\ * & * & \theta^2_{9,3} \end{bmatrix}, \mathbf{\Pi} = \begin{bmatrix} \textcircled{0} & * & * \\ * & \textcircled{0} & * \\ * & * & \textcircled{0} \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} \theta^4_{1,1} & * & * \\ * & \theta^4_{2,2} & * \\ * & * & \theta^4_{3,3} \end{bmatrix}$$

The circled coefficients are fixed at the designated value and thus will not be estimated in the model, which is in consideration for identification of the model. All the other coefficients are estimated with l_1 norm regularization. The * stands for coefficients that are set to be zero in true data generation, i. e. , the corresponding variables

do not lie in the model for generating data, but still exist in the estimated model. There is no regularization on parameters representing variances and co-variances. The path diagrams for true model I and estimated model I are in 2 and 3.

The residual co-variance matrices are set by

$$\boldsymbol{\epsilon}_1 = \sigma^2 I_p, \;\; \boldsymbol{\epsilon}_2 = \sigma^2 I_q, \;\; \boldsymbol{\Psi} = \sigma^2 I_s. \tag{12}$$

In the true model, coefficients $\theta^1_{\cdot}, \theta^2_{\cdot}, \theta^3_{\cdot}$ follow uniform distribution on the intervals $[-1.5, -0.5] \bigcup [0.5, 1.5]$.

In model II, we only modify the $\boldsymbol{\Pi}$ in model I as following to estimate a more complex model.

$$\boldsymbol{\Pi} = \begin{bmatrix} \textcircled{0} & \theta^3_{1,2} & * \\ * & \textcircled{0} & \theta^3_{2,3} \\ \theta^3_{3,1} & * & \textcircled{0} \end{bmatrix}.$$

$\theta^4_{\cdot}$ follows uniform distribution on the intervals $[-0.8, -0.5] \bigcup [0.5, 0.8]$. These picks guarantee that $I - \boldsymbol{\Pi}$ is invertible.

For selecting optimal hyper parameter, we let λ take an geometric decreasing sequence from 0.5 to 0.01 with length of 30 and then use Information Criteria(BIC) (Schwarz et al., 1978) to select λ. Simulation results in (Jacobucci et al., 2016) suggests the advantage of using BIC for model selection. BIC is defined by

$$-2L(\hat{\boldsymbol{\theta}}) + log(N) * np, \tag{13}$$

where np stands for number of nonzero coefficients in all regularized parameters.

We compare among 7 methods. The proposed gradient sampling for regularized SEM is implemented in R package *SEMGS*, using C++ numerical computation library armadillo (Eddelbuettel & Sanderson, 2014; Sanderson, 2010).

We also compare orthant-based LBFGS method(**lbfgs**), an l_1-norm regularized non-convex optimization method implemented in R package *lbfgs*(Coppola, Stewart, & Okazaki, 2014). Since these methods are all implemented in R using C++ to evaluate function values, the comparison is quite fair.

In our simulation we choose the same starting values for all methods, and set the maximum times of iteration=2000. The number of replications is 100. Among different replications the coefficients are different following the distributions mentioned earlier. We summarize on the average of 3 statistics on parameter estimations, false positive rate(FPR), true positive rate(TPR) and mean square error MSE to evaluate the

performance of parameter estimation of different methods. FPR is defined by

$$\text{FPR} = \frac{\#\text{zero coefficients estimated to be nonzero}}{\#\text{zero coefficients}}.$$

TPR is defined by

$$\text{TPR} = \frac{\#\text{nonzero coefficients estimated nonzero}}{\#\text{nonzero coefficients}}.$$

And MSE is the mean square error of the estimated parameters and their true value, including those unpenalized parameters. Methods in the package *regsem* can give an estimation that is not in the domain of the objective function, or giving an implied covariance matrix that is not positive definite in rare cases. In searching for optimal λ we drop these improper estimation and select the λ that gives minimum BIC in the remaining models. Using our package *SEMGS*, all results give proper estimation for all hyper parameters in all cases of the simulation experiment. Two types of penalties, Lasso and SCAD penalty are considered. We fix a=3. 7 in (**??**)suggested by (Fan & Li, 2001). The results are summarized in tables 1-8.

We see from the simulation, gradient sampling can give estimations that have lower FPRs than other methods, in all the settings. For TPR, all these methods give TPRs above 0.8 in all the cases, or close to 1 in some settings, except for lbfgs which has lower TPRs. Note that lbfgs only applies to Lasso penalty. The result on TPR agrees with that in (Jacobucci et al., 2016), where TPR is not shown in the simulation study since they are all close to 1. Gradient sampling has slightly higher TPRs than other methods from *regsem*. For MSE, gradient sampling outperforms other methods. The MSE of estimation with gradient sampling is smaller than other methods. In summary, we conclude in the simulation study gradient sampling gives better estimation of the parameters.

4.2. Real data experiment

In this section we apply regularized SEM to an empirical data experiment. The task is to predict human tissue-specific gene networks using regularized SEM. Recent research has shown effectiveness of using SEM for gene network inference (Cai, Bazerque, & Giannakis, 2013; Liu, de La Fuente, & Hoeschele, 2008). We consider predicting gene network structure via the regularized SEM. The gene expression data is assumed to have a normal distributed measurement noise. The expression level of each gene in the network is treated as the observed variable in the measurement model. Each observed variable measures one latent variable, which represents the de-noised expression level of the gene. The coefficients for latent variables in the measurement model are fixed to be 1, in consideration for model identification. Then the de-noised expression level

is regressed on all other genes. There is no exogenous latent variable in the model.

Specifically, the measurement model is

$$\mathbf{x} = \boldsymbol{\eta} + \boldsymbol{\epsilon}, \tag{14}$$

and the structural model is

$$\boldsymbol{\eta} = \boldsymbol{\Pi}\boldsymbol{\eta} + \boldsymbol{\epsilon}_0. \tag{15}$$

$\boldsymbol{\Pi}$ will indicate the network structure. The diagonal elements of $\boldsymbol{\Pi}$ are all zeros. If the estimated coefficient $\boldsymbol{\Pi}_{ij}$ or $\boldsymbol{\Pi}_{ji}$ is nonzero, then there will be an edge between $\boldsymbol{\eta}_i$ and $\boldsymbol{\eta}_j$, otherwise there will be no edge.

We use RNA-seq gene TPM data downloaded from GTEx data portal[1] (Carithers et al., 2015). A sub-network of 15 genes are considered in human lung tissue. The dataset contains 427 samples. To make a reference network structure, we obtain the network structure predicted by HumanBase[2] (Greene et al., 2015; Krishnan et al., 2016). HumanBase builds tissue specific gene interaction networks by integrating a collection of experimentally verified data sets covering more than ten thousand publications. The edges in the predicted network with a confidence level above a threshold will be showed. The higher the threshold, the sparser the network. Since the network is still generated by data-driven algorithms, we use it only as an approximation of the underlying network.

Figure 4 shows the network predicted by HumanBase as a reference of the true underlying network when the threshold is set at 0.60 and 0.35. The network only contain edges in the network with high confidence given by HumanBase and might omit some of the existing edges between genes. It is reasonable for an algorithm to detect more edges beween the genes, and should also detect at least some of the edges suggested by HumanBase.

Since in simulation study we see the proximal coordinate descent(**pcd**) method outperforms other methods in most of the cases, we only make comparisons between the proposed method **gradsp** and **pcd**. Figures 5 to 8 show the result of the prediction by using these two methods.

We consider different two settings of hyper parameter and penalty types. Comparing to two reference networks, for **gradsp** the FPR is lower and its TPR is higher , which suggests that using **gradsp** the regularized SEM can predict network structures that are more closer to the reference network by HumanBase. Again we point that since the true network structure is intractable, there might be edges yet to be detected by experimental evidence. So here we should care more about the TPR. In the cases

[1] https://www.gtexportal.org/home/datasets

[2] http://hb.flatironinstitute.org/

considered here, the **gradsp** has higher TPR, which means a higher proportion of connections are detected.

4.3. Discussion

In the simulation and real data experiment we see that different optimization methods for a same problem can give different results. This agrees with our aforementioned analysis.

The hyper parameter in the model can also affect the output of the model. Usually the hyper parameter can be estimated from data, for example, by model fitness criteria, or cross validation. In the simulation, we use the same criteria and find different methods will select different hyper parameters. The hyper parameter can also be assigned by the user to obtain a desired sparsity level. In real data experiment we fix the hyper parameter and compare their parameter estimation. In both settings, the proposed method outperforms other methods.

We also observe that for other methods, like **pcdl**, and **pg**, there are extraordinarily large MSE of the parameter estimation in table 5 and 6. This suggests that these methods are not numerically stable, and can give estimated parameters might diverge to points faraway from the true parameters values. The proposed method has a more robust estimation in all the cases in the simulation.

We consider Lasso penalty and SCAD penalty for sparsity. More penalties in regularized SEM can also be solved by gradient sampling, because its convergence requirement can be satisfied by a wide family of functions. Examples of penalties are MCP (Zhang et al., 2010), l_q penalty (Grasmair, Haltmeier, & Scherzer, 2008), fused Lasso (Tibshirani, Saunders, Rosset, Zhu, & Knight, 2005) and group Lasso (Meier, Van De Geer, & Bühlmann, 2008).

5. Conclusion

We explore the current optimization methods for solving regularized SEM. Since the objective function is both non-convex and non-smooth, current optimization methods might not have enough convergence guarantees. We propose to use gradient sampling to solve regularized SEM, so that the algorithm will have theoretical convergence property. We do computer simulation on two lisrel models with varying coefficients and the result shows that using gradient sampling the estimation of the coefficients are closer to the true values than other implementation of the optimization methods. We also apply our method to a gene network structure inference problem and the results show the advantageous performance of the proposed method. An R package implementing gradient sampling for regularized SEM is available on the Internet.

References

Asl, A., & Overton, M. L. (2017). Analysis of the gradient method with an armijo-wolfe line search on a class of nonsmooth convex functions. *arXiv preprint arXiv:1711.08517*.

Asparouhov, T., & Muthén, B. (2009). Exploratory structural equation modeling. *Structural equation modeling: a multidisciplinary journal*, *16*(3), 397–438.

Attouch, H., Bolte, J., Redont, P., & Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of Operations Research*, *35*(2), 438–457.

Attouch, H., Bolte, J., & Svaiter, B. F. (2013). Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, *137*(1-2), 91–129.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, *2*(1), 183–202.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Bollen, K. A. (1989). *Structural equations with latent variables, new york 1989.* John Wiley Interscience.

Bolte, J., Sabach, S., & Teboulle, M. (2014). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, *146*(1-2), 459–494.

Boţ, R. I., Csetnek, E. R., & László, S. C. (2016). An inertial forward–backward algorithm for the minimization of the sum of two nonconvex functions. *EURO Journal on Computational Optimization*, *4*(1), 3–25.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Burke, J., Lewis, A., & Overton, M. (2006). *Hanso (hybrid algorithm for non-smooth optimization): a matlab package based on bfgs, bundle and gradient sampling methods.* Version.

Burke, J. R., Lewis, A. S., & Overton, M. L. (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, *15*(3), 751–779.

Burke, J. V., Lewis, A. S., & Overton, M. L. (2002). Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, *27*(3), 567–584.

Cai, X., Bazerque, J. A., & Giannakis, G. B. (2013). Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations. *PLoS computational biology*, *9*(5), e1003068.

Carithers, L. J., Ardlie, K., Barcus, M., Branton, P. A., Britton, A., Buia, S. A., ... others (2015). A novel approach to high-quality postmortem tissue procurement: the gtex project. *Biopreservation and biobanking*, *13*(5), 311–319.

Coppola, A., Stewart, B., & Okazaki, N. (2014). lbfgs: Limited-memory bfgs optimization [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=lbfgs` (R package version 1.2.1)

Curtis, F. E., & Overton, M. L. (2012). A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. *SIAM Journal on Optimization*, *22*(2), 474–500.

Curtis, F. E., & Que, X. (2013). An adaptive gradient sampling algorithm for non-smooth optimization. *Optimization Methods and Software*, *28*(6), 1302–1324.

Eddelbuettel, D., & Sanderson, C. (2014). Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics & Data Analysis*, *71*, 1054–1063.

Fan, J., & Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, *96*(456), 1348–1360.

Grasmair, M., Haltmeier, M., & Scherzer, O. (2008). Sparse regularization with l_q penalty term. *Inverse Problems*, *24*(5), 055020.

Greene, C. S., Krishnan, A., Wong, A. K., Ricciotti, E., Zelaya, R. A., Himmelstein, D. S., ... et al. (2015). Understanding multicellular function and disease with human tissue-specific networks. *Nature genetics*, *47*(6), 569.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., & Tibshirani, R. (2009). *The elements of statistical learning* (2nd ed.). Springer.

Hastie, T., Tibshirani, R., & Wainwright, M. (2015). *Statistical learning with sparsity: the Lasso and generalizations*. CRC press.

Huang, P.-H. (n.d.). lsl: Latent structure learning [Computer software manual]. Retrieved from `https://rdrr.io/cran/lsl/` (R package version 0.5.6)

Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A penalized likelihood method for structural equation modeling. *psychometrika*, *82*(2), 329–354.

Jacobucci, R. (2017). regsem: Regularized structural equation modeling. *arXiv preprint arXiv:1703.08489*.

Jacobucci, R., Grimm, K. J., Brandmaier, A. M., & Serang, S. (n.d.). regsem: Regularized structural equation modeling [Computer software manual]. Retrieved from `https://cran.r-project.org/web/packages/regsem/index.html` (R package version 1.1.0)

Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized structural equation modeling. *Structural equation modeling: a multidisciplinary journal*, *23*(4), 555–566.

Kaplan, D. (2000). *Structural Equation Modeling: Foundations and Extensions*. SAGE.

Kiwiel, K. C. (2007). Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, *18*(2), 379–388.

Krishnan, A., Zhang, R., Yao, V., Theesfeld, C. L., Wong, A. K., Tadych, A., ... et al. (2016). Genome-wide prediction and functional characterization of the genetic basis of autism spectrum disorder. *Nature neuroscience*, *19*(11), 1454.

Lee, S.-Y. (2007). *Structural equation modeling: A bayesian approach* (Vol. 711). John Wiley & Sons.

Liu, B., de La Fuente, A., & Hoeschele, I. (2008). Gene network inference via structural equation modeling in genetical genomics experiments. *Genetics*, *178*(3), 1763–1776.

Marsh, H. W., Morin, A. J., Parker, P. D., & Kaur, G. (2014). Exploratory structural equation modeling: An integration of the best features of exploratory and confirmatory factor analysis. *Annual review of clinical psychology*, *10*, 85–110.

Meier, L., Van De Geer, S., & Bühlmann, P. (2008). The group Lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*(1), 53–71.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization 2nd*. Springer.

Ochs, P., Chen, Y., Brox, T., & Pock, T. (2014). ipiano: Inertial proximal algorithm for

nonconvex optimization. *SIAM Journal on Imaging Sciences*, *7*(2), 1388–1419.

Parikh, N., Boyd, S., et al. (2014). Proximal algorithms. *Foundations and Trends® in Optimization*, *1*(3), 127–239.

Petersen, K. B., Pedersen, M. S., et al. (2008). *The matrix cookbook* (Vol. 7) (No. 15).

Sanderson, C. (2010). Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments.

Schwarz, G., et al. (1978). Estimating the dimension of a model. *The annals of statistics*, *6*(2), 461–464.

Shor, N. Z. (2012). *Minimization methods for non-differentiable functions* (Vol. 3). Springer Science & Business Media.

Tang, C.-m., Liu, S., Jian, J.-b., & Li, J.-l. (2014). A feasible sqp-gs algorithm for nonconvex, nonsmooth constrained optimization. *Numerical Algorithms*, *65*(1), 1–22.

Tao, P. D., et al. (2005). The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of operations research*, *133*(1-4), 23–46.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., & Knight, K. (2005). Sparsity and smoothness via the fused Lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(1), 91–108.

Tseng, P., & Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, *117*(1-2), 387–423.

Wolfe, P. (1975). A method of conjugate subgradients for minimizing nondifferentiable functions. In *Nondifferentiable optimization* (pp. 145–173). Springer.

Xu, Y., & Yin, W. (2013). A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, *6*(3), 1758–1789.

Zhang, C.-H., et al. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, *38*(2), 894–942.

Zou, H. (2006). The adaptive Lasso and its oracle properties. *Journal of the American statistical association*, *101*(476), 1418–1429.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.

6. Appendix

Table 1. N=200, Lasso penalty, Model I

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 161	0. 285	0. 878	0. 281	0. 301	0. 206
TPR	0. 999	0. 999	0. 987	1	1	0. 233
MSE	0. 0337	0. 0395	0. 2986	0. 0406	0. 0367	0. 5758

Table 2. N=600, Lasso penalty, Model I

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 162	0. 205	0. 815	0. 21	0. 214	0. 291
TPR	1	1	0. 991	1	1	0. 349
MSE	0. 0141	0. 0185	0. 2796	0. 015	0. 017	0. 6542

Table 3. N=200, SCAD penalty, Model I

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 04	0. 091	1	0. 074	0. 095	na
TPR	0. 999	0. 983	1	0. 99	0. 978	na
MSE	0. 0114	0. 0215	0. 379	0. 0169	0. 0238	na

Table 4. N=600, SCAD penalty, Model I

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 006	0. 044	1	0. 035	0. 056	na
TPR	1	0. 996	1	0. 999	0. 993	na
MSE	0. 0038	0. 0104	0. 6753	0. 0068	0. 0122	na

Table 5. N=200, Lasso penalty, Model II

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 395	0. 541	0. 854	0. 554	0. 656	0. 441
TPR	0. 993	0. 987	0. 961	0. 983	0. 985	0. 504
MSE	0. 0339	0. 0509	11. 001	22. 1488	0. 0996	2. 5163

Table 6. N=600, Lasso penalty, Model II

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 412	0. 471	0. 838	0. 488	0. 65	0. 494
TPR	1	0. 994	0. 972	0. 991	0. 989	0. 555
MSE	0. 0152	0. 3594	0. 7122	6923. 404	0. 067	1. 8242

Table 7. N=200, SCAD penalty, Model II

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 114	0. 308	1	0. 302	0. 342	na
TPR	0. 94	0. 877	0. 999	0. 875	0. 874	na
MSE	0. 0284	0. 1815	0. 6037	0. 1531	0. 2644	na

Table 8. N=600, SCAD penalty, Model II

method	**gradsp**	**pcd**	**pcdl**	**pg**	**pcdm**	**lbfgs**
FPR	0. 048	0. 271	1	0. 291	0. 343	na
TPR	0. 975	0. 917	1	0. 92	0. 917	na
MSE	0. 0114	1. 8886	2. 5395	1. 9445	0. 357	na

Table 9. summary of the FPR and TPR for the referenced network with confidence threshold 0. 60

penalty	Lasso				SCAD			
λ	0. 906		0. 676		0. 906		0. 676	
method	**gradsp**	**pcd**	**gradsp**	**pcd**	**gradsp**	**pcd**	**gradsp**	**pcd**
FPR	0. 244	0. 163	0. 407	0. 43	0. 233	0. 291	0. 372	0. 57
TPR	0. 474	0. 421	0. 789	0. 368	0. 526	0. 421	0. 789	0. 579

Table 10. summary of the FPR and TPR for the referenced network with confidence threshold 0. 35

penalty	Lasso				SCAD			
λ	0. 906		0. 676		0. 906		0. 676	
method	**gradsp**	**pcd**	**gradsp**	**pcd**	**gradsp**	**pcd**	**gradsp**	**pcd**
FPR	0. 224	0. 179	0. 343	0. 478	0. 209	0. 269	0. 313	0. 537
TPR	0. 395	0. 263	0. 711	0. 316	0. 421	0. 395	0. 684	0. 632

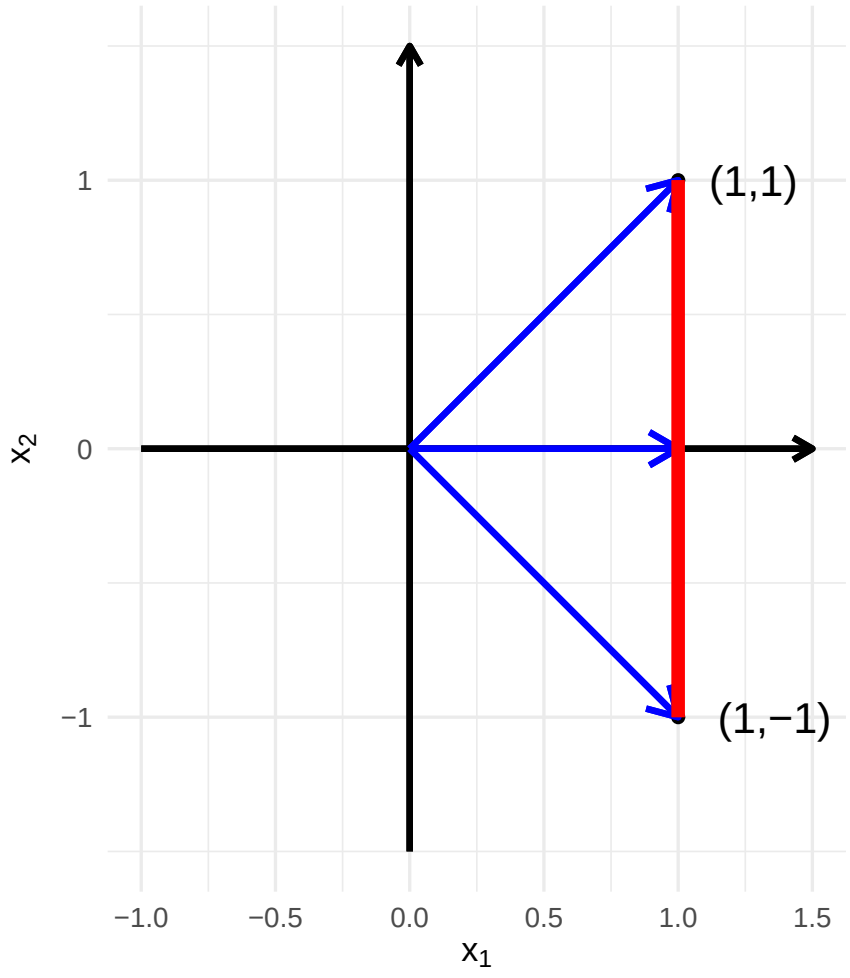


Figure 1. The red line is cv(1, 1), (1, -1), and (1, 0) is point closest to the origin

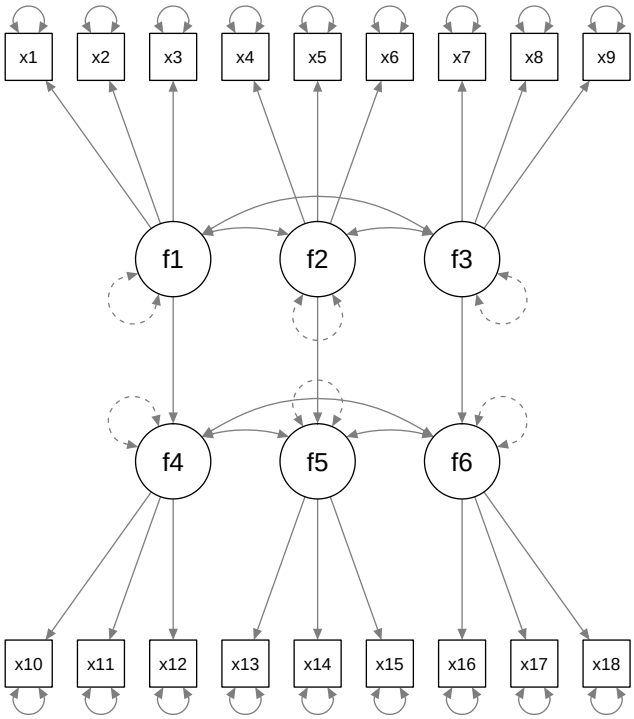


Figure 2. Path plot for true model I

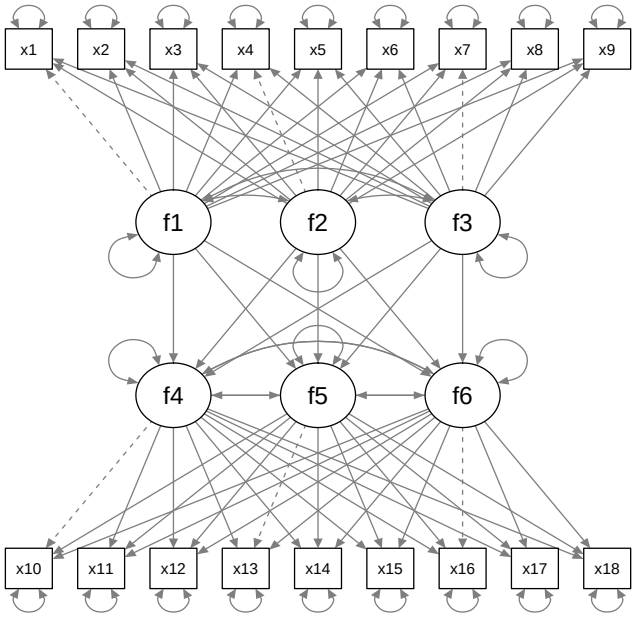


Figure 3. Path plot for estimated model I. Dashed lines represent the coefficients that are fixed to their true value 1 to ensure model identification

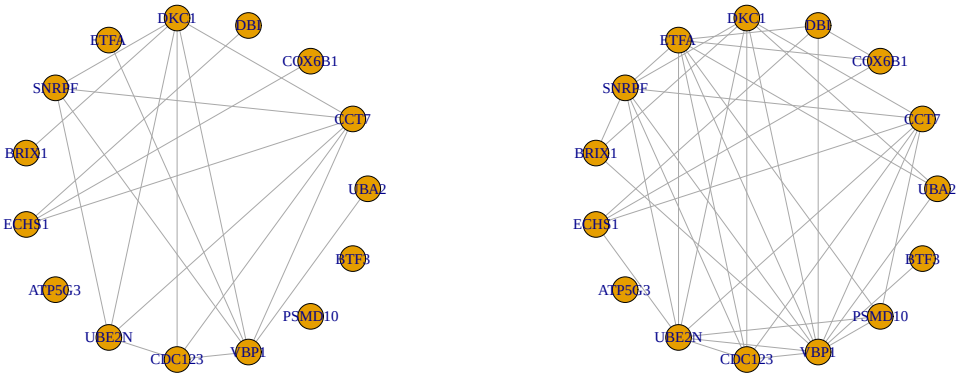
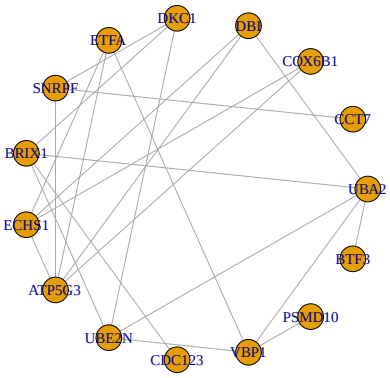


Figure 4. The network predicted by HumanBase. Left:confidence threshold 0.60. Right:confidence threshold 0.35

method:pcd $\lambda = 0.906$ penalty: LASSO



method:gradsp $\lambda = 0.906$ penalty: LASSO

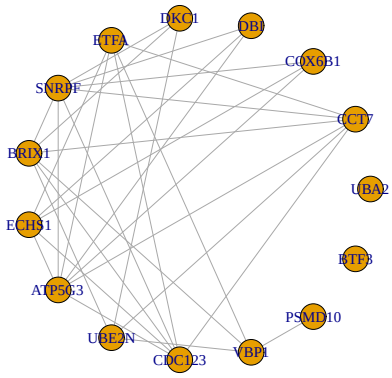
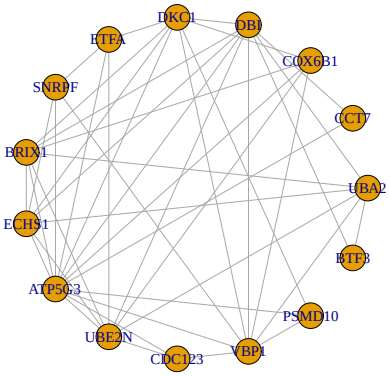


Figure 5.

method:pcd $\lambda = 0.676$ penalty: LASSO



method:gradsp $\lambda = 0.676$ penalty: LASSO

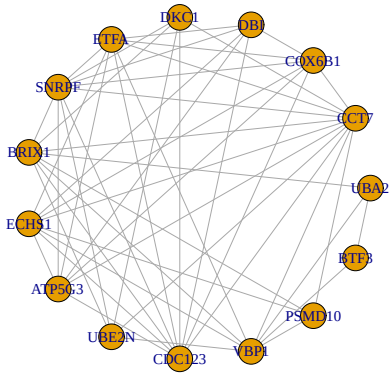
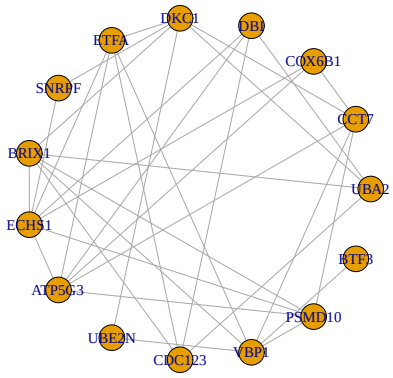


Figure 6.

method:pcd $\lambda = 0.906$ penalty: SCAD



method:gradsp $\lambda = 0.906$ penalty: SCAD

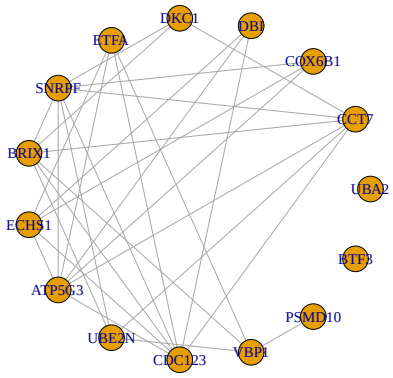
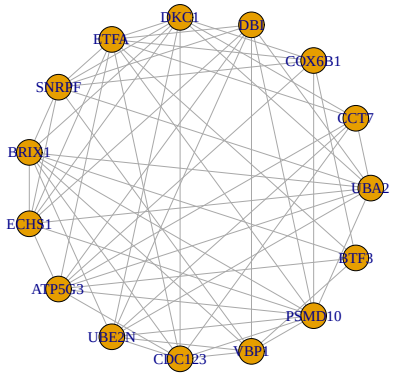


Figure 7.

method:pcd $\lambda = 0.676$ penalty: SCAD



method:gradsp $\lambda = 0.676$ penalty: SCAD

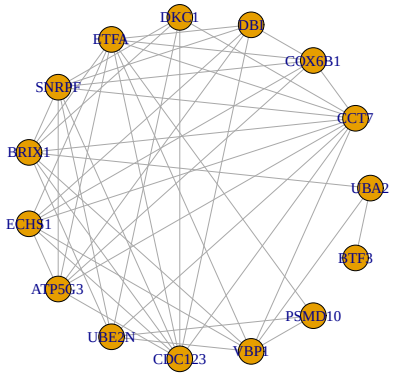


Figure 8.