Submitted to the Annals of Statistics

PARTIAL CORRELATION SCREENING FOR ESTIMATING LARGE PRECISION MATRICES, WITH APPLICATIONS TO CLASSIFICATION

By Shiqiong Huang[†], Jiashun Jin[†], and Zhigang Yao[‡]

Carnegie Mellon University[†] *and National University of Singapore*[‡]

Given n samples $X_1, X_2, \ldots, X_n$ from $N(0, \Sigma)$, we are interested in estimating the $p \times p$ precision matrix $\Omega = \Sigma^{-1}$; we assume Ω is sparse in that each row has relatively few nonzeros.

We propose *Partial Correlation Screening (PCS)* as a new row-by-row approach. To estimate the i-th row of Ω, $1 \leq i \leq p$, PCS uses a *Screen* step and a *Clean* step. In the Screen step, PCS recruits a (small) subset of indices using a stage-wise algorithm, where in each stage, the algorithm updates the set of recruited indices by adding the index j that has the largest empirical partial correlation (in magnitude) with i, given the set of indices recruited so far. In the Clean step, PCS re-investigates all recruited indices, removes false positives, and uses the resultant set of indices to reconstruct the i-th row.

PCS is computationally efficient and modest in memory use: to estimate a row of Ω, it only needs a few rows (determined sequentially) of the empirical covariance matrix. PCS is able to execute an estimation of a large Ω (e.g., $p = 10K$) in a few minutes.

Higher Criticism Thresholding (HCT) is a recent classifier that enjoys optimality, but to exploit its full potential, we need a good estimate of Ω. Note that given an estimate of Ω, we can always combine it with HCT to build a classifier (e.g., HCT-PCS, HCT-glasso).

We have applied HCT-PCS to two microarray data sets ($p = 8K$ and $10K$) for classification, where it not only significantly outperforms HCT-glasso, but also is competitive to the Support Vector Machine (SVM) and Random Forest (RF). These suggest that PCS gives more useful estimates of Ω than the glasso; we study this carefully and have gained some interesting insight.

We show that in a broad context, PCS fully recovers the support of Ω and HCT-PCS is optimal in classification. Our theoretical study sheds interesting light on the behavior of stage-wise procedures.

1. Introduction.

There is always the story of "four blind men and the elephant" [2]. A group of blind men were asked to touch an elephant to learn what it is like. Each one touched a different part, but only one part (e.g., the

[†]Supported in part by NSF Grant DMS-1208315.

AMS 2000 subject classifications: Primary 62H30, 62H20; secondary 62G08, 62P10.

Keywords and phrases: Feature selection, forward and backward selection, glasso, partial correlation, Random Forest, Screen and Clean, sparsity, Support Vector Machine.

tusk, the ear, or the leg). They then compared notes and learnt that they were in complete disagreement, until the King pointed out to them: "All of you are right. The reason that every one of you is telling it differently is because each one of you touched the different part of the elephant. So actually the elephant has all the features you mentioned".

There are several similarities between the elephant tale and the problem on estimating large precision matrices (some are obvious, but some are not).

- Both deal with something enormous: an elephant or a large matrix.
- Both encourage *parallel computing*: either with a group of blind men or a cluster of computers. Individuals only communicate with a 'center' (a king, a master computer), but do not communicate with each other.
- Both are *modest in memory use*. If we are only interested in a small part of the elephant (e.g., the tail), we do not need to scan the whole elephant. If we are only interested in a row of a sparse precision matrix, we don't need to use the *whole* empirical covariance matrix.

Modesty in memory use is especially important when we only have a modest computing platform, where it is easy to hit the memory ceiling.

Given a data matrix $X \in \mathbb{R}^{n,p}$. We write

$$X = [x_1, x_2, \ldots, x_p] = [X_1, X_2, \ldots, X_n]',$$

For simplicity, we assume each row X_i are zero mean Gaussian vectors:

$$X_i \overset{iid}{\sim} N(0, \Sigma), \qquad \text{where } \Sigma \in \mathbb{R}^{p,p} \text{ is positive definite.} \tag{1.1}$$

Denote by $\hat{\Sigma}$ by the empirical covariance matrix

$$\hat{\Sigma}(i,j) = (x_i, x_j)/n. \tag{1.2}$$

The precision matrix

$$\Omega = \Sigma^{-1}, \tag{1.3}$$

is unknown to us but is presumably sparse, in the sense that each row of Ω has relatively few nonzeros, and the primary interest is to estimate Ω.

Our primary interest is in the 'large n, really large p' regime [36], where it is challenging to estimate Ω precisely with energy-efficient computing.

The glasso [43] is a well-known approach which estimates Ω by optimizing the ℓ^1-penalized objective function of the log-likelihood associated with $\hat{\Sigma}$. The glasso is not exactly modest in memory use, and for large p (e.g., $p = 10K$), the glasso can be unsatisfactorily slow, especially when the tuning parameter is small. Also, by its design, it is unclear how to implement the glasso with extensively parallel computing. The modified glasso[1] [40] improves the performance of glasso but only partially solves these problems.

Alternatively, we can estimate Ω row by row, with Nearest Neighborhood (NN) [12], scaled-lasso (slasso) [37], and CLIME [10] being the examples. These methods relate the problem of estimating an individual row of Ω to a linear regression model and apply some variable selection approaches: NN, slasso and CLIME apply the lasso, scaled-lasso, and Dantzig Selector respectively. Unfortunately, for $p \geq 10K$, these methods are unsatisfactorily slow, simply because the lasso, scaled-lasso, and Dantzig Selector are not fast enough to accomplish p different variable selections in a timely fashion. They are not exactly modest in memory use either: to estimate a row of Ω, they need to use the whole matrix of $\hat{\Sigma}$ or X.

We propose *Partial Correlation Screening (PCS)* as a new approach to estimating the precision matrix. PCS has the following appealing features.

- *Allowing for energy-efficient computing.* PCS estimates Ω row by row using a fast screening algorithm, and is able to estimate Ω for $p = 10K$ or larger in a timely manner on a modest computing platform.
- *Modesty in memory use.* To estimate each row of Ω, PCS does not need the whole matrix of $\hat{\Sigma}$. It only needs the diagonals and a few rows (selected sequentially) of $\hat{\Sigma}$, if Ω is sufficiently sparse. This enables us to bypass the RAM limit and to accommodate much larger Ω.

However, we must note that, practically, estimating Ω is *rarely* the ultimate goal. In many applications, the goal is usually to use the estimated Ω to improve statistical inference (e.g., classification, multiple testing).

Data Name	Source	n (# of subjects)	p (# of genes)
Rats	Yousefi et al. (2010)	181	8491
Liver	Yousefi et al. (2010)	157	10237

In this paper, largely motivated by interests in gene microarray, we focus on how to use the estimated precision matrix to improve classification results with microarray data. Table 1 displays two microarray data sets we study in this paper. In each data set, we have samples from two classes (e.g., normal versus diseased), and each sample is measured over the same set of genes. The main interest is to use the data set to construct a trained classifier.

[1]The modified glasso assumes the glasso solution is a block-diagonal matrix so that we can split the original glasso problem into many smaller size problems that can be solved separately. Specifically, it checks whether the estimated glasso solution is block diagonal for a given tuning parameter (usually large), resulting a set of uncorrelated variables. If so, then one can simply apply the glasso to each block separately, leading to speed improvements, provided that the number of uncorrelated variables is large.

We propose to combine PCS with the recent classifier of Higher Criticism Thresholding (HCT) [13, 22], and to build a new classifier HCT-PCS. In [13, 22], they investigated a two-class classification setting with a Gaussian graphical model. Assuming samples from two classes share the same sparse precision matrix Ω, they showed that, given a reasonably good estimate of Ω, HCT enjoys optimal classification behaviors. The challenge, however, is to find an algorithm that estimates the precision matrix accurately with energy-efficient computation; this is where PCS comes in.

We apply HCT-PCS to the two data sets above. In these data sets, the precision matrix is unknown, so it is hard to check whether PCS is more accurate for estimating Ω than existing procedures. However, the class labels are given, which can be used as the 'ground truth' to evaluate the performance of different classifiers. We find that

- HCT-PCS significantly outperforms other versions of HCT (say, HCT-glasso, where Ω is estimated by the glasso[2]), suggesting that PCS yields more accurate estimates of Ω than other approaches (the glasso, say).
- HCT-PCS is competitive, in both computation time (especially when n is large) and classification errors, to the more popular classifiers of Support Vector Machine (SVM) [9] and Random Forest (RF) [7].

1.1. *PCS: the idea.* We present the key idea of PCS, leaving the formal introduction to Section 1.2. To this end, we consider an *idealized* case where we are allowed to access all 'small-size' principal sub-matrices of Σ (but not any 'large-size' sub-matrices), and study how to use such sub-matrices to reconstruct Ω. Since any small-size principal sub-matrix of Σ can be well-approximated by the corresponding sub-matrix of $\hat{\Sigma}$ (despite that $\hat{\Sigma}$ as a whole is a bad approximation to Σ due to $p \gg n$), once we understand such an idealized case, we know how to deal with the real one.

Write $\Omega = (\omega_1, \omega_2, \ldots, \omega_p)$ so that ω_i' is the i-th row. Fixing $1 \le i \le p$, we wish to understand what could be a reasonable approach to reconstructing ω_i' using only 'small-size' sub-matrices of Σ. Define

$$S^{(i)}(\Omega) = \{1 \le j \le p : \ \omega_i(j) \neq 0, j \neq i\}. \tag{1.4}$$

Note that $\{i\} \cup S^{(i)}(\Omega)$, not $S^{(i)}(\Omega)$, is the support of ω_i.

Definition 1.1 *For any matrix $A \in \mathbb{R}^{n,p}$ and subsets $\mathcal{I} = \{i_1, i_2, \ldots, i_M\} \subset \{1, \ldots, n\}$ and $\mathcal{J} = \{j_1, \ldots, j_K\} \subset \{1, \ldots, p\}$, $A^{\mathcal{I},\mathcal{J}}$ denotes the $M \times K$ sub-matrix such that $A^{\mathcal{I},\mathcal{J}}(m, k) = A(i_m, j_k)$, $1 \le m \le M, 1 \le k \le K$ (indices in either $\mathcal{I}$ or $\mathcal{J}$ are not necessarily arranged in the ascending order).*

[2]We remark that it is the modified glasso that is implemented throughout the paper, but we still refer it as glasso only for convenience.

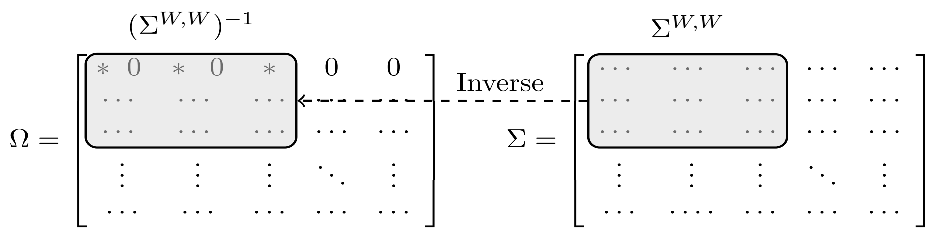

Fig 1. *The first row of Ω only has nonzeros at column $1, 3, 5$, marked with "$*$". For any subset W such that $\{1, 3, 5\} \subset W$, the first rows of $\Omega^{W,W}$ and $(\Sigma^{W,W})^{-1}$ are the same.*

Here is an interesting observation. For any subset W such that

$$(\{i\} \cup S^{(i)}(\Omega)) \subset W \subset \{1, 2, \dots, p\}, \tag{1.5}$$

we can reconstruct ω_i' by only knowing a specific row of $(\Sigma^{W,W})^{-1}$!

Lemma 1.1 *If (1.5) holds and index i is the k-th index in W, then the k-th rows of $\Omega^{W,W}$ and $(\Sigma^{W,W})^{-1}$ are equal, though generally $\Omega^{W,W} \neq (\Sigma^{W,W})^{-1}$.*

The proof of Lemam 1.1 is elementary so we omit it; see also Figure 1. Lemma 1.1 motivates a two-step *Screen and Clean* approach (an idea for variable selection that is applicable in many cases [21, 30, 31, 32, 39]).

- In the Screen stage, we identify a subset $S_*^{(i)} = S_*^{(i)}(\Sigma, p)$, in hopes of $S^{(i)}(\Omega) \subset S_*^{(i)}$.
- In the Clean stage, we reconstruct ω_i' from the matrix Σ^{W_*,W_*} following the idea in Lemma 1.1, where $W_* = \{i\} \cup S_*^{(i)}$.

Seemingly, the key is how to screen. Our proposal is to use the *partial correlation*, a concept closely related to the precision matrix [8]. Consider an (ordered) subset $W \subset \{1, 2, \dots, p\}$ where i and j are the first and the last indices, respectively. Let $S = W \setminus \{i, j\}$. For any vector $Z \sim N(0, \Sigma)$, the partial correlation between $Z(i)$ and $Z(j)$ given $\{Z(k) : k \in S\}$ is defined as

$$\rho_{ij}(S) = \frac{-1 \cdot \text{first row last column of } \left(\Sigma^{W,W}\right)^{-1}}{\left[\text{product of the first and last diagonals of } \left(\Sigma^{W,W}\right)^{-1}\right]^{1/2}}. \tag{1.6}$$

By Lemma 1.1 and Lemma 2.2 (to be introduced), $S^{(i)}(\Omega) \subset (\{i\} \cup S) \Longleftrightarrow \rho_{ij}(S) = 0$ for all $j \notin (\{i\} \cup S)$. This motivates a stage-wise algorithm for choosing $S_*^{(i)}$ as follows, where we use the partial correlation to recruit *exactly one node* in each step before the algorithm terminates.

Initialize with $S_0^{(i)} = \emptyset$*. Suppose the algorithm has run* $(k-1)$ *steps and has not yet stopped. Let* $S_{k-1}^{(i)} = \{j_1, j_2, \ldots, j_{k-1}\}$ *be all the nodes recruited (in that order) by far. In the* k*-th step, if* $\rho_{ij}(S_{k-1}^{(i)}) \neq 0$ *for some* $j \notin (\{i\} \cup S_k^{(i)})$*, let* $j = j_k$ *be the index with the largest value of* $|\rho_{ij}(S_{k-1}^{(i)})|$*, and update with* $S_k^{(i)} = S_{k-1}^{(i)} \cup \{j_k\}$*. Otherwise, terminates and let* $S_*^{(i)} = S_{k-1}^{(i)}$.

It is shown in Theorem 2.1 that under mild conditions, the algorithm terminates at $\leq C|S^{(i)}(\Omega)|^2$ steps, at which point, $S^{(i)}(\Omega) \subset S_*^{(i)}$ and $\rho_{ij}(S_*^{(i)}) = 0$ for all $j \notin (\{i\} \cup S_*^{(i)})$. Letting $W_* = \{i\} \cup S_*^{(i)}$, we can then use Σ^{W_*, W_*} to reconstruct ω_i', following the connection given in Lemma 1.1.

Since $\hat{\Sigma}^{W,W} \approx \Sigma^{W,W}$ as long as $|W|$ is small, the idea above is readily extendable to the 'real case', provided that Ω is sufficiently sparse. This idea is fleshed out in Section 1.2, where PCS is formally introduced.

1.2. *PCS: the procedure.* When we invert a principle sub-matrix of $\hat{\Sigma}$, it is frequently desirable to use ridge regularization. Fixing $\delta > 0$, for any positive definite matrix W, define the *Ridge Regularized Inverse* by

$$\mathcal{I}_\delta(W) = \begin{cases} W^{-1}, & \text{if all eigenvalues of } W \geq \delta, \\ (W + \delta I_{|W|})^{-1}, & \text{otherwise}, \end{cases} \tag{1.7}$$

where I_k denotes the $k \times k$ identity matrix (we may drop "k" for simplicity).

For any indices i, j and subset $S \subset \{1, 2, \ldots, p\} \setminus \{i, j\}$, let $W = \{i\} \cup S \cup \{j\}$ and suppose i and j are the first and last indices in the subset. Introduce the *regularized empirical partial correlation* by

$$\hat{\rho}_{ij}^{(\delta)}(S) = \frac{-1 \cdot \text{first row last column of } \mathcal{I}_\delta(\hat{\Sigma}^{W,W})}{\left[\text{product of the first and last diagonals of } \mathcal{I}_\delta(\hat{\Sigma}^{W,W})\right]^{1/2}}.^{3} \tag{1.8}$$

PCS is row-by-row method specifically designed for large Ω where it may be impossible to load the whole matrix $\hat{\Sigma}$ to the software (e.g., Matlab on a laptop) due to RAM limit. Fortunately, to estimate a row of Ω, PCS only needs a few rows of $\hat{\Sigma}$ determined squentially. Therefore, we can first deposit $\hat{\Sigma}$ in a 'data center',[4] and then only load the rows of $\hat{\Sigma}$ that we need (separately for estimating different rows of Ω) to the software. See Figure 2.

Fix $1 \leq i \leq p$, a small number $\delta > 0$, a properly large integer $L > 0$, and a tuning parameter $q > 0$. Let

$$t_q^* = t_q^*(p, n) = q\sqrt{2\log(p)/n}. \tag{1.9}$$

To estimate row i of Ω, PCS consists of four steps (also see Figure 2).

[3] If $\delta = 0$ and $\hat{\Sigma}$ is replaced by Σ, then $\hat{\rho}_{ij}^{(\delta)}(S)$ reduces to $\rho_{ij}(S)$ defined in (1.6).

[4] A 'data center' can be many things: a hard disk of a laptop, or a large data depository.

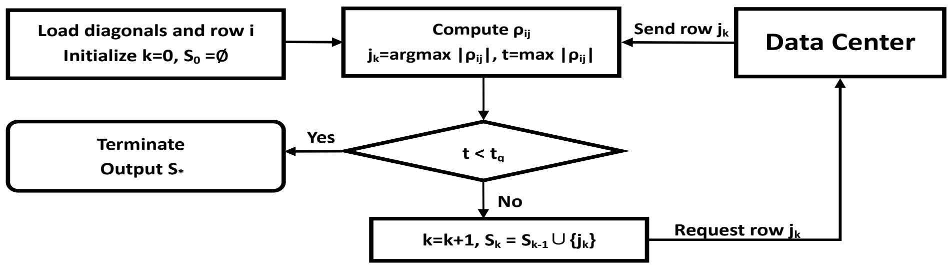

FIG 2. *Flow chart of PCS (short-hand notations are used for simplicity)*

- *Initial step.* Let $\hat{S}_0^{(i)} = \emptyset$, and load the i-th row and the diagonals of $\hat{\Sigma}$ (from the data center to the software; same below).
- *Screen step.* Suppose the algorithm has not yet terminated at the end of step $(k-1)$, and let $\hat{S}_{k-1}^{(i)} = \{j_1, j_2, \ldots, j_{k-1}\}$ be all the nodes recruited so far (in that order). If

  $$k < L, \quad \text{and} \quad |\hat{\rho}_{ij}^{(\delta)}(\hat{S}_{k-1}^{(i)})| \geq t_q^* \text{ for some } j \notin (\{i\} \cup \hat{S}_k^{(i)}), \tag{1.10}$$

  let $j = j_k$ be the node satisfying $j_k = \text{argmax}_{\{j: j \notin (\{i\} \cup \hat{S}_{k-1}^{(i)})\}} |\hat{\rho}_{ij}^{(\delta)}(\hat{S}_{k-1}^{(i)})|$ (pick the smallest index if there are ties). Load the j_k-th row of $\hat{\Sigma}$ to the software and update $\hat{S}_k^{(i)}$ by $\hat{S}_k^{(i)} = \hat{S}_{k-1}^{(i)} \cup \{j_k\} = \{j_1, j_2, \ldots, j_k\}$. Otherwise, the algorithm terminates, and set $\hat{S}_*^{(i)} = \hat{S}_*^{(i)}(t, X; p, n)$ as $\hat{S}_{k-1}^{(i)}$ (indices are arranged in the order they are recruited).
- *Clean step.* Let $\hat{\eta}'$ be the first row of $\mathcal{I}_\delta(\hat{\Sigma}^{\hat{W}_*, \hat{W}_*})$, where $\hat{W}_* = \{i\} \cup \hat{S}_*^{(i)}$. Write $\hat{W}_* = \{i, j_1, j_2, \ldots, j_k\}$ (nodes arranged in that order). Denote the set of selected nodes after cleaning by $\hat{S}_{**}^{(i)} = \hat{S}_{**}^{(i)}(t, X; p, n) = \{j_\ell : |\hat{\eta}(\ell+1)| \geq t_q^*, 1 \leq \ell \leq k\}$. Let $\hat{W}_{**} = \{i\} \cup \hat{S}_{**}^{(i)}$ (where i is the first node) and let $A = \mathcal{I}_\delta(\hat{\Sigma}^{\hat{W}_{**}, \hat{W}_{**}})$. Estimate the i-th row of Ω by

  $$\hat{\Omega}^*(i,j) = \begin{cases} \text{first row } \ell\text{-th column of } A, & j \text{ is the } \ell\text{-th node in } \hat{W}_{**}, \\ 0, & j \notin \hat{W}_{**}. \end{cases}$$

- *Symmetrization.* $\hat{\Omega}^{pcs} = [\hat{\Omega}^* + (\hat{\Omega}^*)']/2$[5].

PCS has three tuning parameters (q, δ, L), but its performance is not sensitive to different choices of (δ, L), as long as they are in a reasonable

[5]Alternatively, one may symmetrize $\hat{\Omega}^*$ by minimizing $\|\Omega - \hat{\Omega}^*\|_*$ up to that Ω is symmetric and $supp(\Omega) \subset \{(i,j) : j \in \hat{S}_{**}(i) \text{ or } i \in \hat{S}_{**}(j)\}$. Here, $\|\cdot\|_*$ is an appropriate matrix norm; this is similar to that in [42] where $\|\cdot\|_*$ is taken to be the matrix ℓ^1-norm.

range. In this paper, we set $(\delta, L) = (.1, 30)$, so essentially PCS only has one tuning parameter q. In practice, how to set q is generally a difficult problem. Our primary focus on real data analysis is classification, in which settings we select q by cross validations. See Section 1.4 for details.

The computation cost of PCS is $O(p^2L^3 + np^2)$, where $O(np^2)$ is the cost of obtaining $\hat{\Sigma}$ from the data matrix X, and the L^3 term comes from the step of sequentially inverting matrices of sizes $2, 3, \ldots, L+1$. Also, PCS estimates Ω row by row and allows for parallel computing. Together, these make PCS a fast algorithm that can have energy-efficient computing for large precision matrices. For example, with $(q, \delta, L) = (.2, .1, 30)$, it takes the PCS only about 5 and 7.5 minutes on the rats and the liver data sets, respectively.

PCS is also modest in memory use: to estimate a row of Ω, PCS only needs the diagonals and no more than L rows of $\hat{\Sigma}$. This enables PCS to bypass the RAM limit even if p is large. Note that some communication costs between the software and 'data center' are expected, but these seem unavoidable when p is large. For other methods (e.g., the glasso, CLIME), one usually needs the whole $\hat{\Sigma}$ even if we only wish to estimate a single row of Ω; for these methods, it is unclear how to deal with the RAM limit.

1.3. *Applications to classification.* Consider a classification setting where we have samples $(\tilde{X}_i, Y_i)$, $1 \leq i \leq n$, from two classes, where $\tilde{X}_i \in \mathbb{R}^p$ are the feature vectors and $Y_i \in \{-1, 1\}$ are the class labels. Given a fresh sample $\tilde{X} \in \mathbb{R}^p$ where the associated class label $Y \in \{-1, 1\}$ is unknown, the goal is to use $(\tilde{X}_i, Y_i)$ to construct a trained classifier and to use it to predict Y.

We model $\tilde{X}_i$ with a Gaussian graphical model, where for two distinct mean vectors $\mu^{\pm} \in \mathbb{R}^p$ and a covariance matrix $\Sigma \in \mathbb{R}^{p,p}$,

$$\tilde{X}_i \sim N(\mu^{\pm}, \Sigma), \qquad \text{if } Y_i = \pm 1, \text{ respectively.} \tag{1.11}$$

Similar to that of (1.3), we assume the precision matrix $\Omega = \Sigma^{-1}$ is sparse, in the same sense. Additionally, let μ be the contrast mean vector:

$$\mu = \mu^+ - \mu^-; \tag{1.12}$$

we assume μ is sparse in that only a small fraction of its entries is nonzero.

We are primarily interested in classification for microarray data, in which context, model (1.11) is frequently used; see for example Efron [17] and Fan *et al.* [22]. Note that for simplicity, we assume the covariance matrices associated with two classes are equal; see Section 4 for more discussions on this. The matrix Σ captures the correlations among the measurement noise, and plays an important role in classification. For the two data sets in Table 1, model (1.11) might deviate from the ground truth, but the good thing is that PCS is not tied to model (1.1) and our proposed classifier works quite well on these data sets; see Section 1.4.

Higher Criticism Thresholding (HCT) is a recent classifier proposed in [13, 22], which adapts Fisher's Linear Discriminant Analysis (LDA) to the modern regime of 'large n, really large p'. In the idealized case where Ω is known or can be estimated reasonably well, HCT is shown to have optimal classification behaviors for model (1.11). The question is then how to estimate Ω accurately with energy-efficient computing.

We consider three approaches to estimating Ω: PCS, the glasso [40], and FoBa (i.e., the classical forward-backward variable selection method [35]).[6] In the literature, FoBa has not yet been proposed as an approach to estimating Ω; here, we propose FoBa as a variant of PCS. See Section 1.6.

To apply PCS, the glasso, or FoBa, it is more convenient to start with the empirical correlation matrix $\hat{R}$ (see below) than with $\hat{\Sigma}$. Let n_1 and n_2 be the sample sizes of Class 1 and Class 2, let $\hat{\mu}^{\pm} \in \mathbb{R}^p$ be the sample mean vectors for Class 1 and Class 2, respectively, and let $\hat{s}^{\pm} \in \mathbb{R}^p$ be the vectors of sample standard deviations for class 1 and class 2, respectively. The pooled standard deviation associated with feature j is then

$$\hat{s}(j) = \sqrt{[(n_1-1)(s^+(j))^2 + (n_2-1)(s^-(j))^2]/(n_1+n_2-2)}. \tag{1.13}$$

For $i = 1, 2, \ldots, n$, let $\hat{\mu}_i^* \in \mathbb{R}^p$ be the vectors satisfying $\hat{\mu}_i^* = \hat{\mu}^+$ if $i \in$ Class 1 and $\hat{\mu}_i^* = \hat{\mu}^-$ otherwise. The empirical correlation matrix $\hat{R} \in \mathbb{R}^{p,p}$ is then

$$\hat{R}(j,k) = (n\hat{s}(j)\hat{s}(k))^{-1} \sum_{i=1}^{n} (\tilde{X}_i(j) - \hat{\mu}_i^*(j))(\tilde{X}_i(k) - \hat{\mu}_i^*(k)). \tag{1.14}$$

Once $\hat{R}$ is obtained, we apply each of the three methods (PCS, glasso, FoBa) and denote the estimates by $\hat{\Omega}^{pcs}$, $\hat{\Omega}^{glasso}$, and $\hat{\Omega}^{foba}$.

For $\hat{\Omega}$ being either of the three estimates, the corresponding HCT-classifier (denoted by HCT-PCS, HCT-glasso, and HCT-FoBa) consists of the following steps for classification.

- Let $Z \in \mathbb{R}^p$ be the vector of *summarizing* t*-scores*: $Z(j) = (\hat{\mu}^+(j) - \hat{\mu}^-(j))/(n_0 \cdot \hat{s}(j))$, $1 \le j \le p$, where $n_0 = (n_1^{-1} + n_2^{-1})^{1/2}$.
- Normalize Z by $Z^*(j) = (Z(j) - u(j))/d(j)$, where $u(j)$ and $d(j)$ are the mean and standard deviation of different entries of Z.
- Apply the *Innovated Transformation* [22]: $\tilde{Z} = \hat{\Omega}Z^*$.

[6]CLIME and scaled-lasso [10, 37] are not included for comparison, as they are unsatisfactorily slow for $p \ge 8K$. The thresholding approach by Bickel and Levina [5] is not included either, for it focuses on the case where Σ is sparse (but Ω may be non-sparse).

- *Threshold choice by Higher Criticism.* For each $1 \leq j \leq p$, obtain a P-value by $\pi_j = P(|N(0,1)| \geq (\hat{\Omega}(j,j))^{-1/2}|\tilde{Z}(j)|)$. Sort the P-values ascendingly by $\pi_{(1)} < \pi_{(2)} < \ldots < \pi_{(p)}$. Let $\hat{j}$ be the index among the range $1 \leq j \leq \alpha_0 p$ and that maximizes the so-called HC functional $HC_{p,j} = [j/p - \pi_{(j)}]/\sqrt{(1-j/p)(j/p)}$ for all j in the range of $1 \leq j \leq \alpha_0 p$ (we usually set $\alpha_0 = .2$, as suggested by [13]). The HC threshold $t_p^{HC} = t_p^{HC}(\tilde{Z}, \hat{\Omega}, p, n)$ is the magnitude of the $\hat{j}$-th largest entry (in magnitude) of $\tilde{Z}$.
- *Assign weights by thresholding.* Let $w_{HC}(j) = \mathrm{sgn}(\tilde{Z}(j)) \cdot 1\{|\tilde{Z}(j)| \geq t_p^{HC}\}$, $1 \leq j \leq p$. Denote $w_{HC} = (w_{HC}(1), w_{HC}(2), \ldots, w_{HC}(p))'$.
- *Classification by post-selection LDA.* We normalize the test feature $\tilde{X}$ by $\tilde{X}^*(j) = [\tilde{X}(j) - (\hat{\mu}^+(j) + \hat{\mu}^-(j))/2]/\hat{s}(j)$, $1 \leq j \leq p$. Let $L_{HC}(\tilde{X}) = (w_{HC})'\hat{\Omega}\tilde{X}^*$. We classify $Y = \pm 1$ according to $L_{HC}(\tilde{X}) \gtrless 0$.

The rationale behind step 2 is the phenomenal work by Efron [16] on empirical null. Efron found that for microarray data, there is a substantial gap between the (marginal) distribution of the theoretical null and that of the empirical null, and it is desirable to bridge the gap by renormalization. This step is specifically designed for microarray data, and may not be necessary for other types of data (say, simulated data). Also, note that when normalizing the test feature $\tilde{X}$, we use $(\hat{\mu}^{\pm}, \hat{s})$ which do not depend on $\tilde{X}$.

1.4. *Comparison: classification errors with microarray data.* We consider the two gene microarray data sets in Table 1. The original rats data set was collected in a study on gene expressions of live rats in response to different drugs and toxicants, and we use the cleaned version by [41]. The data set consists of 181 samples measured on the same set of 8491 genes, where 61 samples are labeled as toxicants, and the other 120 as other drugs. The original liver data set was collected in a study on the hepatocellular carcinoma (HCC), and we also use the cleaned version by [41]. The data set consists of 157 samples measured on the same set of 10,237 genes, 82 of them are tumor samples, and the other 75 non-tumor.

We consider a total of 6 different classifiers: naive HCT (where we pretend that Ω is diagonal and apply HCT without estimating off-diagonal of Ω; denoted by nHCT), HCT-PCS, HCT-glasso, HCT-FoBa, and two popular classifiers: Support Vector Machine (SVM) and Random Forest (RF).

Among them, nHCT is tuning free, three methods have one tuning parameter: λ for HCT-glasso, 'cost' for SVM, and 'number of trees' for RF. The tuning parameter for HCT-glasso (and also those of HCT-PCS and HCT-FoBa) come from the method of estimating the precision matrix. HCT-PCS has three tuning parameters (δ, L, q), but it is relatively insensitive to (δ, L).

In this paper, we set $(\delta, L) = (0.1, 30)$, so PCS only have one tuning parameter q. For HCT-FoBa, we use the package by [45], which has three tuning parameters: a back fitting parameter (set by the default value of .5 here), a ridge regression parameter $\delta > 0$ and a step size parameter L. As parameters (δ, L) have similar roles to (δ, L) in PCS, we set them as $(\delta, L) = (0.1, 30)$. The performance of either PCS or FoBa is relatively insensitive to different choices of (δ, L), as long as they fall in a certain range.

In our study, we use two layers of 3-fold data splitting. To differentiate one from the other, we call them the *data-splitting* and *cv-splitting*. The former is for comparing classification errors of different methods across different data splitting and it is particularly relevant to evaluating the performance on real data, while the latter is for selecting tuning parameters. The latter is not required for nHCT or HCT-FoBa.

- *Data-splitting.* For each data set, we apply 3-fold random splitting to the samples in either of the two classes (25 times, independently).
- *Cv-splitting.* For each resultant training set from the data splitting, we apply 3-fold random splitting to the samples in either of the two classes (25 times, independently).

Sample indices for the 25 data splittings and sample indices of the 25 cv-splittings associated with each of the data splitting can be found at `www.stat.cmu.edu/~jiashun/Research/software`.

We now discuss how to set the tuning parameters in HCT-PCS, HCT-glasso, SVM, and RF. For HCT-PCS, the tuning parameter is q. A relatively small q could lead to over fitting and slow computation, and a relatively large q would result in low precision. We find that the interesting range for q is $0.05 \leq q \leq 0.5$. We discretize this interval evenly with an increment of 0.05. The increment is sufficiently small, and a finer grid does not have much difference. For each data splitting, we determine the best q using 25 independent cv-splitting, picking the one that has the smallest "cv testing error". This q value is then plugged into HCT-PCS for classification

For HCT-glasso, the interesting range for the parameter λ is $0.8 \leq \lambda \leq 1$. Since the algorithm starts with empirical correlation matrix, it is unnecessary to go for $\lambda > 1$ (the resultant estimate would be a scalar times the $p \times p$ identity matrix, a simple result of the KKT condition [23]). On the other hand, it is very time consuming by taking $\lambda < 0.8$. For example, if we take $\lambda = 0.65, 0.7$, and 0.75, then on a $12GB$ RAM machine, it takes the glasso more than a month for $\lambda = 0.65$, about a month for $\lambda = 0.7$, and about 180 hours for $\lambda = 0.75$ to complete all 25×25 combinations of data-splitting and cv-splitting, correspondingly. Similar to that of HCT-PCS, for each data splitting, we take $\lambda \in \{0.8, 0.85, 0.9, 0.95, 1\}$ and use the 25 cv-splittings to decide the best λ, which is then plugged into HCT-glasso for classification.

For SVM, we use the package from http://cran.r-project.org/web/packages/e1071/index.html. We find the interesting range for the 'cost' parameter is between 0.5 and 5, so we take 'cost' to be $\{0.5, 1, 1.5, \dots, 5\}$ and use cv-splitting to pick the best one. For RF, we use the package downloaded from http://cran.r-project.org/web/packages/randomForest/index.html. RF has one tuning parameter 'number of trees'. We find that the interesting range for 'number of trees' is between 50 to 500, so we take it to be $\{50, 100, \dots, 500\}$ and use cv-splitting to decide the best one.

The average classification (testing) error rates of all 6 methods across 25 different data splittings are in Table 1. The standard deviations of the errors are relatively large, due to the large variability in data splitting. For more informative comparison, we present the number of testing errors associated with all 25 data splittings in Figure 3 (rats data) and Figure 4 (liver data), respectively, where the 25 data splittings are arranged in a way so that the corresponding errors of HCT-PCS are increasing from left to right.

TABLE 1
Comparison of classification errors (average for 25 data-splitting). The error rates and their standard deviations (in brackets) are reported in percentage (e.g., 5.7 means 5.7%).

Data	HCT-PCS	HCT-FoBa	HCT-glasso	nHCT	SVM	RF
Rats	5.7(3.05)	8.6(3.36)	20.3 (5.10)	15.1(6.49)	6.9(4.02)	13.5(3.99)
Liver	4.2(3.60)	10.0(4.64)	20.2 (6.38)	10.5(4.30)	3.5(2.72)	4.3(3.00)

From the left panel of Figure 3, we see that for the rats data, HCT-glasso, nHCT and RF are all above HCT-PCS. To better show the differences among HCT-PCS, HCT-FoBa and SVM, we further plot the testing errors in the right panel. Figure 4 provides the similar information for liver data, with HCT-PCS, SVM and RF being highlighted in the right panel. The results suggest: for rats data, HCT-PCS outperforms all methods with the average errors, including SVM and RF; for liver data, HCT-PCS is slightly inferior to SVM, but still outperforms all other methods; for both data sets, HCT-PCS significantly outperforms all other HCT-based methods (nHCT, HCT-glasso, HCT-FoBa), which further suggests PCS gives a better estimate for the precision matrix than the glasso and FoBa.

The computation time is hard to compare, as it depends on many factors such as the data-splitting in use, how professional the code is written, and how capable the user handles the computation. Therefore, the complexity comparison summarized in Table 2 can only be viewed as a qualitative one (for the complexity of the glasso, see [33]). We run all methods using Matlab,

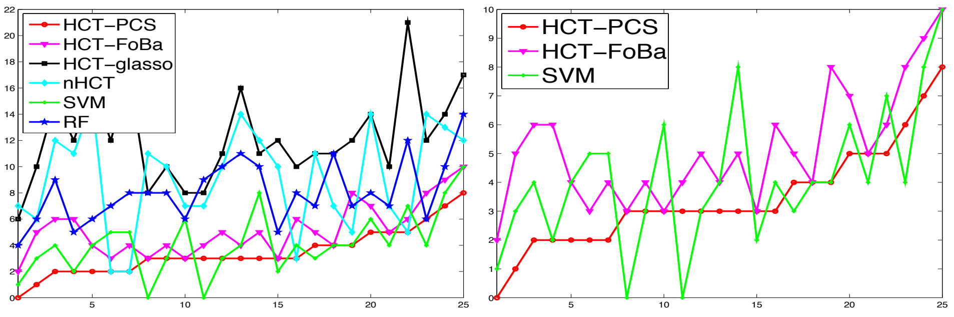

FIG 3. *Comparison of testing errors for the rats data. Left: number of errors (y-axis) of 6 methods for 25 data splittings (x-axis; labels of different rounds of data splitting are arranged in a way so that the errors of HCT-PCS increase from left to right). Right: the same information but only with HCT-PCS, HCT-FoBa and SVM (for a better view).*

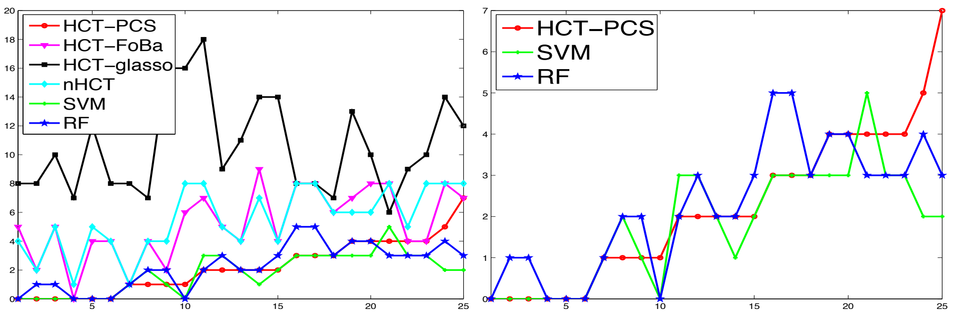

FIG 4. *Comparison of testing errors for the liver data. Left: number of errors (y-axis) of 6 methods at 25 data splittings (x-axis; labels of different rounds of data splitting are arranged in a way so that the errors of HCT-PCS increase from left to right). Right: the same information but only with HCT-PCS, SVM and RF (for a better view).*

with an exception of FoBa, SVM and RF by R, on a workstation with 8 CPU cores and 12GB RAM, and the real time elapsed in computation is recorded upon one data splitting.

TABLE 2
Comparison of computational complexity of all classifiers and running time for rats and liver data at a single data splitting. Here, n is the number of samples, p is the number of variables and T is the number of trees used in RF (assuming $p \geq n$ and $L^3 \leq O(n)$; computation times are based on 25 cross validations for all but HCT-FoBa and nHCT). For HCT-FoBa, the computation time accounts for no cross validation.

	HCT-PCS	HCT-FoBa	HCT-glasso	nHCT	SVM	RF
Complexity	$O(np^2)$	$O(np^2)$	$O(p^3)$–$O(p^4)$	$O(np\log(p))$	$O(n^2p)$	$O(Tnp\log(n))$
Time (rats)	166.8 min	8.7 min	380 min	0.11 min	7.7 min	21.5 min
Time (liver)	241.0 min	12.9 min	890 min	0.12 min	7.7 min	26.3 min

It is noteworthy for much larger n (e.g., $n = 2000$), SVM becomes much slower, showing a disadvantage of SVM, compared to PCS, FoBa, and the glasso. See Section 3 (simulation section) for settings with much larger n.

Remark. Both PCS and FoBa use ridge regularization, but PCS uses it on an as-needed basis (see (1.7)) and FoBa uses it more conventionally. In Table 3, we compare the classification errors of PCS and FoBa for the cases of with ($\delta = .1$) and without ridge regularization ($\delta = 0$). The results suggest a substantial improvement by using ridge regularization. On the other hand, we find that the classification errors for both methods are relatively insensitive to the choice of δ, as long as they fall in an appropriate range. In this paper, we choose $\delta = .1$ for all real data experiments.

TABLE 3
Comparison of classification errors for HCT-PCS and HCT-FoBa for $\delta = .1$ and $\delta = 0$ (e.g., 5.7 means 5.7%; numbers in the brackets: standard deviations).

Data	HCT-PCS ($\delta = .1$)	HCT-PCS ($\delta = 0$)	HCT-FoBa ($\delta = .1$)	HCT-FoBa ($\delta = 0$)
Rats	5.7(3.05)	13.2(3.57)	8.6(3.36)	13.7(5.10)
Liver	4.2(3.60)	12.5(6.08)	10.0(4.64)	15.1(7.48)

1.5. *Comparison with glasso over the estimated* Ω. That fact that HCT-PCS significantly outperforms HCT-glasso and HCT-FoBa in classifications suggests that PCS may give a 'better' estimation of Ω than FoBa and the glasso. We now compare the estimated precision matrices by the glasso, PCS, and FoBa with the two data sets. Figure 5 presents the histograms of the number of nonzeros of different rows in $\hat{\Omega}$ for the three different methods. For all histograms, we use the whole data set without data splitting. For PCS, we use $(q, \delta, L) = (0.2, 0.1, 30)$. For the glasso, we use $\lambda = 0.8$. For FoBa, we use $(\delta, L) = (0.1, 30)$ so that it is consistent with PCS. The histograms look similar when we change the tuning parameters in the appropriate range. Figure 5 reveals very different patterns of the estimated Ω.

- For most rows, glasso estimates all off-diagonals to be 0. For some of the rows, the glasso estimate can have a few hundreds of nonzeros.
- For either of the PCS and the FoBa estimates, the number of nonzeros in each row can be as large as a few ten's, but no smaller than 10.

While the ground truth is unknown, it seems that the estimates by PCS or FoBa make more sense: it is hard to believe that the off-diagonals of Ω are all 0 for most of the rows; it is more likely that in most of the rows, we have at least a few nonzeros. Partially, this explains why the classification errors of the glasso is the largest among the three methods. It also explains why the naive HCT has unsatisfactory behaviors (recall that in nHCT, we pretend that Ω is diagonal).

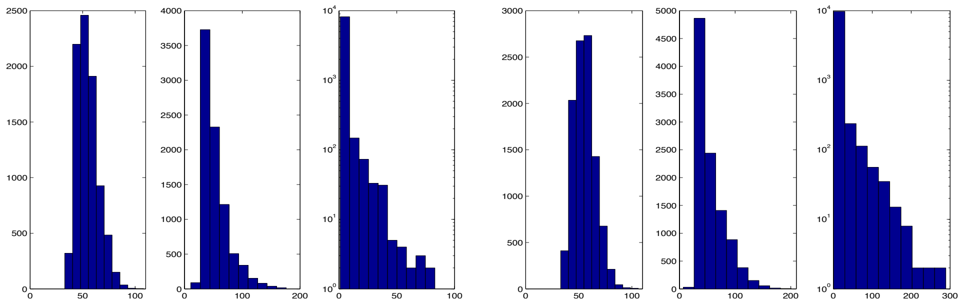

FIG 5. *Panels 1-3: histograms of the number of nonzeros in different rows of* $\hat{\Omega}$ *for the rats data by PCS, FoBa, and the glasso (y-axis in Panel 3 is* $\log(\#$*of nonzeros*$))$*. For PCS and FoBa, the number of nonzeros range from* 33 *to* 108 *and from* 11 *to* 176*, respectively. For the glasso, in* 7550 *out of* 8491 *rows, all off-diagonals are estimated as* 0 *(maximal number of nonzeros in a row is* 83*). Panels 4-6: similar but are for the liver data.*

Huge [46] and Quic [25] are two improved versions of the glasso package [23]. We find these two packages have comparable computing speed to PCS when p is moderately small (e.g., $p \leq 4000$) and the precision matrix is very sparse (e.g., .1% nonzero entries). For the two microarray data sets, the dimension p is much larger, and the precision matrix is much less sparse. For these two data sets, we find that Huge and Quic are significantly slower than PCS. At the beginning, we try to implement Huge and Quic with the platform in the previous section (8 CPU cores, 12GB RAM) to complete the task, but find that it takes rather long time to complete the task (for all 25 data splittings and 25 cross validations). Eventually, we decide to use a better platform (56 CPU cores, 120GB RAM), and we estimate that it takes Huge and Quic 7.964 and 42.1705 days to complete the task, respectively. Also, in Table 2, if we replace the glasso package by the Huge package, we get very similar classification errors; this is not very surprising as they are essentially the same method, but with different implementations.

1.6. *Comparison with FoBa over the estimated* Ω. Forward and Backward regression (FoBa) is a classical approach to variable selection, which is proposed by Draper and Smith as early as 1960's [35]. FoBa can be viewed as an extension of the classical Forward Selection (FS) procedure [35], where the difference is that FoBa allows for backward elimination, but FS does not. FS and FoBa have been studied carefully recently (e.g., [15, 44, 45]).

To the best of our knowledge, FS and FoBa have not yet been proposed as an approach to estimating the precision matrix, but we can always develop them into such an approach as follows. Fix $1 \leq i \leq p$. Recall that $X' = [x_1, x_2, \ldots, x_p]$ and that $\Omega = [\omega_1, \omega_2, \ldots, \omega_p]$. It is known that we can always associate each row of Ω with a linear regression model as follows [8]:

$$x_i = (\omega_i(i))^{-1} \sum_{j \neq i} \omega_i(j) x_j + z_i, \qquad z_i \sim N(0, \sigma^2 \cdot I_n), \tag{1.15}$$

where $\sigma^2 = 1/\omega_i(i)$ and $z_i = x_i - (\omega_i(i))^{-1} \sum_{j \neq i} \omega_i(j) x_j$ is independent of $\{x_j : j \neq i\}$. We can then apply either FS or FoBa to (1.15) for each $1 \leq i \leq p$, and symmetrize the whole matrix in the same way as the last step of PCS; the resultant procedure is an approach to estimating Ω. A small gap here is that, for each $1 \leq i \leq p$, FS and FoBa attempt to estimate the vector $(\omega_i(i))^{-1}\omega_i$, not ω_i itself (as we desire).

This is closely related to PCS, but differs in several important ways. Since FoBa is viewed as an improvement over FS, we only compare PCS with FoBa.

The most obvious difference between PCS and FoBa is that, in their 'forward selection' steps, the objective function for recruiting new nodes are different. PCS uses the partial correlation (1.8), and FoBa uses the correlation between x_j and the residuals. The following lemma elaborates two objective functions and is proved in the supplemental material [26].

Lemma 1.2 *For i, j, and $S \subset \{1, 2, \ldots, p\}$ such that $i \neq j$, $i, j \notin S$, and $|S| \leq n - 2$, the objective functions in the 'forward selection' steps of PCS and FoBa associated with $\delta = 0$ are well defined with probability 1, equalling*

$$\hat{\rho}_{ij}(S) = x_i'(I - H_S)x_j / \sqrt{x_i'(I - H_S)x_i \cdot x_j'(I - H_S)x_j}, \tag{1.16}$$

and

$$\hat{\rho}^*_{ij}(S) = x_i'(I - H_S)x_j / \|x_j\|, \tag{1.17}$$

respectively, where H_S is the projection from $\mathbb{R}^n$ to the subspace $\{x_k : k \in S\}$.

PCS and FoBa are also different in philosophy. It is well-known that FS tends to select "false variables". For remedy, FoBa proposes "immediate backward elimination": in each step, FoBa is allowed to add or remove one or more variables, in hopes that whenever we falsely select one or more variables, we can remove them immediately. PCS takes a very different strategy. We recognize that, from a practical perspective, the signals are frequently "rare and weak", meaning that Ω is sparse and that nonzero entries are relatively small individually. "Rare and weak" signal is a recent notion in high dimensional data analysis, the definition of which may vary from occurrence to occurrence, but usually it means that signals are hard to identify if we don't know their locations, but relatively easy to identify (if only) when the locations are known to us. See [14, 31] for review on recent researches for "rare and weak" signals. In such cases, "immediate backward elimination" is impossible and we must tolerate many "false discoveries". Motivated by this, PCS employs a Screen and Clean methodology, which attempts to include all the true nodes while keeping the "false discoveries" as few as possible. Our results on the two microarray data sets support the "rare and weak" viewpoint: for example, in Figure 5, the symmetrization step has a significant impact on the histograms of PCS and FoBa for both data sets, which implies that the "false discoveries" are unavoidable.

Though it can be viewed as a method for variable selection, Screen and Clean method has a strong root in the literature of large-scale multiple testing and in genetics and genomics, where the "rare and weak" viewpoint is especially appropriate. In rare and weak settings, Screen and Clean is more appropriate than other variable selection approaches whose focus is frequently on rare and strong signals. See [14, 31] for more discussions.

In practice, the above differences may lead to noticeable differences between the estimates of Ω by PCS and FoBa. To illustrate, we consider the estimation of row #3823 of Ω associated with data splitting #25 of the rats data, and compare how the forward selection steps of PCS ($\delta = 0.1$) and FoBa ($\delta = 0.1$) are different from each other. The cleaning step of PCS and the backward selection of FoBa are omitted for comparison.

- **PCS ($\delta = 0.1$).** In the Screen step, PCS stops at step 26, and the 26 recruited nodes are: 3823, 8199, 1466, 4164, 6674, 1087, 931, 2419, 5016, 679, 6726, 1059, 5410, 8116, 6183, 1242, 4348, 6492, 147, 5174, 4561, 4096, 2763, 5894, 8140, and 6532.
- **FoBa ($\delta = 0.1$).** We run FoBa for 31 steps. It turns out that 4 of the steps are backward steps (one node deleted in each). The 27 nodes FoBa recruits in each of the forward steps are: 3823, 8199, 4144, 1628, 5707, 931, 1532, 5410, 3620, 2700, 5188, 7933, 2729, 8048, 1212, 2197, 1087, 2337, 5665, 6556, 1962, 8417, 7567, 4164, 1312, 6726, and 4436.

Figure 6 displays the two sets of selected nodes (left panel) by PCS and FoBa and their corresponding coefficients (right panel) given in (1.16) and (1.17), respectively. We see that the first two recruited nodes by PCS and FoBa are the same, corresponding to large coefficients, either in (1.16) and (1.17). All other nodes recruited by PCS and FoBa are different, corresponding to comparably smaller coefficients (either in (1.16) or (1.17)). This suggests a "rare and weak" setting where PCS and FoBa differ significantly from each other. Also, this provides an interesting angle of explaining why PCS outperforms FoBa in terms of classification error.

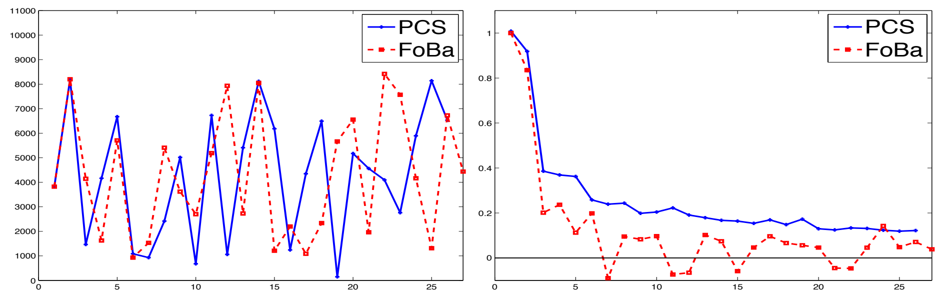

Fig 6. *Left: the set of nodes recruited by PCS (solid) and FoBa (dashed) in the forward steps (PCS:* 26 *nodes; FoBa:* 27 *nodes). Right: objective functions (PCS: (1.16); FoBa: (1.17)) corresponding to the nodes on the left. PCS and FoBa only share the first* 2 *nodes that have the largest (in magnitude) objective functions for PCS and FoBa, respectively.*

1.7. *Summary and contributions.* The most prominent advantage of PCS is that it achieves a good balance between sound theoretical properties and practical feasibility and utility. In theory, we show that PCS yields exact support recovery in many cases where other methods (e.g., CLIME and the glasso) may fail to do so; see Section 2.5. In computation, PCS is faster than many existing approaches (e.g., CLIME, NN, the glasso and its variants Huge and Quic), especially when the precision matrix is large and relatively less sparse. In applications, in addition to a careful comparison of classification error rates, we have also provided interesting light on why some methods behave unsatisfactory. Note that there is no significant gap between the method we use for theoretical analysis and that for applications.

The contribution of the paper is three-fold. First, we propose PCS as a new row-by-row approach to estimating large sparse precision matrices Ω. To estimate each row, we develop a greedy stage-wise algorithm using the empirical partial correlations. PCS is computationally efficient and modest in memory use. These two features enable PCS to execute accurate estimation of the precession matrices with energy-efficient computing, and open doors to accommodating Ω of much larger sizes (e.g., $p \geq 50K$).

Second, we combine PCS with HCT [22] for a new classifier HCT-PCS and apply it successfully to two microarray data sets. HCT-PCS is competitive in classification errors, compared to the more popular classifiers of SVM and RF. HCT-PCS is tuning free (given an estimate of Ω), enjoys theoretical optimality [22], and fully exploits the sparsity in both the feature vectors and the precision matrix. SVM and RF, however, can be unstable with regard to tuning. For example, the tuning parameter in SVM largely relies on training data and structure of the kernel function employed to transform the feature space; this instability of regularization could end up with non-sparse support vectors [4, 11]. SVM and RF are found faster than HCT-PCS in Section 1, but such an advantage is much less prominent for larger n.

HCT-PCS gives more satisfactory classification results than HCT-glasso, suggesting that PCS gives 'better' or 'more useful' estimates for the precision matrix. The glasso is relatively slow in computation when p is as large as $10K$, especially when the tuning parameter is small. For either of two microarray data sets, the glasso estimates are undesirable: in a majority of rows of $\hat{\Omega}$, all off-diagonals are 0. HCT-PCS also gives more satisfactory classification results than HCT-FoBa, and two main differences between PCS and FoBa are (a) PCS and FoBa use very different objective functions in screening, (b) FoBa proposes to remove 'falsely selected nodes' by immediate backward deletion, while PCS adopts a "rare and weak signal" view point, and proposes to keep all 'falsely selected nodes' until the end the Screen step and then remove them in the Clean step.

1.8. *Content and notations.* Section 2 presents the main theoretical results, where we show why and when PCS and HCT-PCS work using a general framework. In particular, in Section 2.5, we show that PCS yields exact support recovery in many settings where other methods (e.g., CLIME and the glasso) may not. Section 3 presents the simulations. Section 4 contains discussions and extensions. Section 5 contains the proofs of two key lemmas. Proofs for theorems and other lemmas are in the supplementary material [26].

In this paper, for any vector a, $\|a\|$ denotes the vector ℓ^2-norm. For any matrix A, $\|A\|$ denotes the matrix spectral norm, $\|A\|_1$ denotes the matrix ℓ^1-norm and $\|A\|_{\max}$ denotes the entry-wise max norm. $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ denote the maximum and minimum eigenvalues of A, respectively. For any matrix $B \in \mathbb{R}^{n,p}$ and two subsets $\mathcal{I}, \mathcal{J}$, $B^{\mathcal{I},\mathcal{J}}$ is the same as in Definition 1.1.

2. Main results.

For simplicity, we only study the version of PCS without ridge regularization, and drop the superscript "(0)" by writing

$$\hat{\rho}_{ij}(S) = \hat{\rho}_{ij}^{(0)}(S), \qquad \text{for any subset } S\text{, random or non-random.}$$

We simply set $L = p$, so PCS has only one tuning parameter q. In this section, $C > 0$ is a generic constant which may vary from occasion to occasion.

Theoretically, to characterize the behavior of PCS, there are two major components: how PCS behaves in the idealized case where we have access to 'small-size' principal sub-matrices of Σ (but not any of the 'large-size' sub-matrices), and how to control the stochastic errors. Below, after some necessary notations, we discuss two components in Sections 2.1-2.2. The main results are presented in the end of Section 2.2.

For any positive definite matrix A, recall that $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ denote the smallest and largest eigenvalues, respectively. For any $1 \leq k \leq p$, define

$$\mu_k^{(1)}(A) = \min_{\{|S|=k\}}\{\lambda_{\min}(A^{S,S})\}, \qquad \mu_k^{(2)}(A) = \max_{\{|S|=k\}}\{\lambda_{\max}(A^{S,S})\}, \tag{2.18}$$

where S is a subset of $\{1, 2, \ldots, p\}$. Also, for an integer $1 \leq K \leq p$, we say that a matrix $A \in \mathbb{R}^{p,p}$ is K-sparse if each row of A has no more than K nonzero off-diagonals. Let $\mathcal{M}_p$ be the set of all $p \times p$ positive definite matrices, let $0 < c_0 \leq 1$ be a fixed constant, and let

$$N = N(p, n) = \text{the smallest integer that exceeds } n/\log(p). \tag{2.19}$$

We consider the following set of Σ (as before, Ω and Σ are tied to each other by $\Omega = \Sigma^{-1}$) denoted by $\mathcal{M}_p^*(s, c_0) = \mathcal{M}_p^*(s, c_0; n)$:

$$\mathcal{M}_p^*(s, c_0) = \{\Sigma \in \mathcal{M}_p\text{: } \Omega \text{ is } s\text{-sparse, } \mu_N^{(1)}(\Sigma) \geq c_0,\ \mu_N^{(2)}(\Sigma) \leq c_0^{-1}\}. \tag{2.20}$$

We use p as the driving asymptotic parameter, so $n \to \infty$ as $p \to \infty$. We allow s (and other parameters below) to depend on p. However, c_0 is a constant not depending on p. Recall that

$$S^{(i)}(\Omega) = \{1 \leq j \leq p : j \neq i, \Omega(i, j) \neq 0\}.$$

Introduce the *minimum signal strength* by

$$\tau_p^* = \tau_p^*(\Sigma) = \min_{1 \leq i \leq p} \tau_p(i), \qquad \text{where} \quad \tau_p(i) = \tau_p(i; \Sigma) = \min_{j \in S^{(i)}(\Omega)}\{|\Omega(i, j)|\}.$$

Definition 2.1 *Fix $1 \leq i \leq p$. We call j a signal node (in row i) if $j \neq i$ and $\Omega(i, j) \neq 0$, and a noise node (in row i) if $j \neq i$ and $\Omega(i, j) = 0$.*

Here, whenever there is no confusion, we may drop the part "in row i". The so-called *Lagging Time* and *Energy At Large (EAL)* plays a key role in characterizing PCS. Suppose we apply PCS to estimate the i-th row of Ω.

Denote the k-th *Selecting Time* for row i by $\hat{m}^{(i)}(k) = \hat{m}^{(i)}(k; X, \Sigma)$, $1 \leq k \leq |S^{(i)}(\Omega)|$; this is the index of the stage at which we select a signal node for the k-th time. By default, $\hat{m}(0) = \hat{m}(0; X, \Sigma) = 0$. The k-th *Lagging Time* for row i is then

$$\hat{\ell}^{(i)}(k) = \hat{\ell}^{(i)}(k; X, \Sigma) = \hat{m}^{(i)}(k) - \hat{m}^{(i)}(k-1) - 1, \qquad 1 \leq k \leq |S^{(i)}(\Omega)|.$$

This is the number of noise nodes PCS recruits between the steps we recruit the $(k-1)$-th and the k-th signal nodes. Additionally, suppose we are now at the beginning of stage m in the Screen step of PCS, and let $\hat{S}^{(i)}_{m-1}$ be the set of all recruited nodes as before. We say a signal node is "At Large" if we have not yet recruited it. The *Energy At Large at stage* m (for row i) is

$$\hat{E}^{(i)}(m) = \hat{E}^{(i)}(m; X, \Sigma) = \sum_{j \in (S^{(i)}(\Omega) \setminus \hat{S}^{(i)}_{m-1})} \Omega(i,j)^2, \qquad 1 \le m \le p-1.$$

In the idealized case when we apply PCS to Σ, Selecting Time, Lagging Time and EAL reduce to their non-stochastic counterparts, denoted correspondingly by

$$m^{(i)}(k) = m^{(i)}(k; \Sigma), \quad \ell^{(i)}(k) = \ell^{(i)}(k; \Sigma), \quad \text{and} \quad E^{(i)}(m) = E^{(i)}(m; \Sigma).$$

Whenever there is no confusion, we may drop the superscript "(i)" for short.

2.1. *Behavior of PCS in the idealized case.* Consider the idealized case where we have access to all 'small-size' principal sub-matrices of Σ. We wish to investigate how the Screen step of PCS behaves. Let $\rho_{ij}(S)$ be as in (1.6). In this idealized case, recall that PCS runs as follows. Initialize with $S^{(i)}_0 = \emptyset$. Suppose the algorithm has run $(m-1)$ steps and has not yet stopped. Let $S^{(i)}_{m-1} = \{j_1, j_2, \ldots, j_{m-1}\}$ be all the nodes recruited (in that order) by far. At stage m, if $\rho_{ij}(S^{(i)}_{m-1}) \neq 0$ for some $j \notin (\{i\} \cup S^{(i)}_{m-1})$, let $j = j_m$ be the index with the largest value of $|\rho_{ij}(S^{(i)}_{m-1})|$, and update with $S^{(i)}_m = S^{(i)}_{m-1} \cup \{j_m\}$. Otherwise, terminate and let $S^{(i)}_* = S^{(i)}_{m-1}$.

The key of the analysis lies in the interesting connection between partial correlations and EAL. The following two lemmas are proved in Section 5.

Lemma 2.1 *Fix p, $0 < c_0 \le 1$, $1 \le i, s \le p$ and $\Sigma \in \mathcal{M}^*_p(s, c_0)$. For each $1 \le k \le |S^{(i)}(\Omega)|$,*

$$\sum_{m^{(i)}(k-1) < m < m^{(i)}(k)} \rho^2_{ij_m}(S^{(i)}_{m-1}) \le \left[\mu^{(2)}_{m^{(i)}(k)+s-k}(\Sigma)\right]^2 \sum_{j \in \left(S^{(i)}(\Omega) \setminus S^{(i)}_{m^{(i)}(k)-1}\right)} \Omega(i,j)^2.$$

Lemma 2.2 *Fix p, $0 < c_0 < 1$, $1 \le i, s \le p$ and $\Sigma \in \mathcal{M}^*_p(s, c_0)$. For each $m \ge 1$,*

$$\sum_{j \in (S^{(i)}(\Omega) \setminus S^{(i)}_{m-1})} \rho^2_{ij}(S^{(i)}_{m-1}) \ge \frac{[\mu^{(1)}_{m+s}(\Sigma)]^3}{\mu^{(2)}_{m+s}(\Sigma)} \sum_{j \in (S^{(i)}(\Omega) \setminus S^{(i)}_{m-1})} \Omega(i,j)^2.$$

Recall that $\Sigma \in \mathcal{M}_p^*(s, c_0)$, so $\mu_{m+s}^{(1)}(\Sigma) \geq C$ and $\mu_{m+s}^{(2)}(\Sigma) \leq C$. Fix $1 \leq k \leq |S^{(i)}(\Omega)|$. Suppose we have recruited $(k-1)$ signals and $(|S^{(i)}(\Omega)| - k + 1)$ ones are at large. The implications of these lemmas are:

- The sum of squares of all such partial correlations associated with noise nodes we recruit between the $(k-1)$-th and the k-th Selecting Times is smaller than a constant C times the EAL associated with the signal nodes that are currently At Large.
- PCS is a greedy algorithm. For each noise node we recruit between the $(k-1)$-th and the k-th Selecting Times, the square of the associated partial correlation is no smaller than that of one of the signal nodes At Large, which in turn is greater than $C(|S^{(i)}(\Omega)| - k + 1)^{-1}$ times the EAL associated with all signal nodes that are currently At Large.
- As a result, the k-th Lagging Time satisfies $\ell^{(i)}(k; \Sigma) \leq C(|S^{(i)}(\Omega)| - k + 1) \leq C(s - k + 1)$, and PCS must have recruited all true signal nodes in no more than $C\sum_{k=1}^{s}(s-k+1) \leq Cs^2$ steps, at which point, all partial correlations are 0 and the algorithm stops immediately.

The above arguments are made precise in the following theorem, the proof of which can be found in the supplementary material [26].

Theorem 2.1 *Suppose $\Sigma \in \mathcal{M}_p^*(s, c_0)$, and $s^2 \log(p) = o(n)$. In the idealized case that we can access all principal sub-matrices of Σ with size no more than $N(p, n)$ defined in (2.19), for each row $1 \leq i \leq p$, the following holds:*

- *At each stage m before all signal nodes are recruited, there exists $j \in (S^{(i)}(\Omega) \setminus S_{m-1}^{(i)})$ such that $|\rho_{ij}(S_{m-1}^{(i)})| \geq C\tau_p^*$, and PCS keeps running.*
- *PCS takes no more than Cs^2 steps to terminate.*
- *When PCS terminates, $\rho_{ij}(S_*^{(i)}) = 0$ for all $j \notin (\{i\} \cup S_*^{(i)})$.*

2.2. *Consistency of PCS.* In this section, we aim to extend Theorem 2.1 to the real case where we have access to small principal sub-matrices of $\hat{\Sigma}$ instead of Σ. Recall that in the Screen step of PCS, we use the threshold

$$t_q^* = t_q^*(p, n) = q \cdot \sqrt{2\log(p)/n}. \tag{2.21}$$

We hope that there is a $q > 0$ such that except for a negligible probability,

- The algorithm stops at no more than Cs^2 steps.
- Suppose we are at stage m of the Screen step of PCS. If the algorithm has not yet recruited all the signal nodes by stage $(m-1)$, then there is a $j \notin (\{i\} \cup \hat{S}_{m-1}^{(i)})$ such that $|\hat{\rho}_{ij}(\hat{S}_{m-1}^{(i)})| \geq t_q^*(p, n)$. If the algorithm has recruited all signal nodes by stage $(m-1)$, then for all $j \notin (\{i\} \cup \hat{S}_{m-1}^{(i)})$, $|\hat{\rho}_{ij}(\hat{S}_{m-1}^{(i)})| < t_q^*(p, n)$.

Such a 'phase transitional' effect ensures PCS to run till all signal nodes are recruited.

The key is to characterize the stochastic fluctuations. Under mild conditions, we can show that except for a probability of $o(p^{-3})$, there is a constant c_1 that only depends on c_0 in (2.20) such that for each $m \geq 1$,

$$\max_{j \notin (\{i\} \cup \hat{S}^{(i)}_{m-1})} |\hat{\rho}_{ij}(\hat{S}^{(i)}_{m-1}) - \rho_{ij}(\hat{S}^{(i)}_{m-1})| \leq c_1 s \sqrt{2\log(p)/n}. \tag{2.22}$$

We need the minimum signal strength to be large enough to counter the effect of stochastic fluctuations. In light of this, we assume

$$\tau_p^*/[s\sqrt{\log(p)/n}] \to \infty, \tag{2.23}$$

$$2c_1 s\sqrt{2\log(p)/n} \leq t_q^*(p,n) \leq (1/2)c_0^2\tau_p^*. \tag{2.24}$$

where c_0 is as in (2.20). The constants 2 and 1/2 are chosen for convenience and can be replaced by any constants $a > 1$ and $b \in (0,1)$, respectively. When (2.23)-(2.24) hold, we are able to derive results similar to those in Lemmas 2.1-2.2, which can then be used to derive the 'phase transitional' phenomenon aforementioned. Roughly saying, with high probability: if all signal nodes have not yet been recruited by stage $(m-1)$, then the partial correlation associated with the next node to be recruited is at least $C\tau_p^* - c_1 s\sqrt{2\log(p)/n}$ which is much larger than the threshold t_q^* and so PCS continues to run. On the other hand, once all signal nodes are recruited, the partial correlation associated with all remaining nodes falls below $c_1 s\sqrt{2\log(p)/n}$ which is no larger than $t_q^*/2$, and PCS stops immediately.

The above arguments are made precise in the following theorem, which is the main result of this paper and proved in the supplementary material [26].

Theorem 2.2 *Fix $1 \leq i \leq p$ and apply the Screen step of PCS to row i. Suppose $\Sigma \in \mathcal{M}_p^*(s, c_0)$, $s^2\log(p) = o(n)$, the minimum signal strength τ_p^* satisfies (2.23), and the threshold $t_q^*(p,n)$ satisfies (2.24) with the constant c_1 properly large. With probability at least $1 - o(p^{-3})$:*

- *At each stage m before all signal nodes are recruited, there exists $j \in (S^{(i)}(\Omega) \setminus \hat{S}^{(i)}_{m-1})$ such that $|\hat{\rho}_{ij}(\hat{S}^{(i)}_{m-1})| \gtrsim c_0^2\tau_p^*$, and PCS keeps running.*
- *PCS takes no more than Cs^2 steps to terminate.*
- *Once PCS recruits the last signal node, it stops immediately, at which point, $|\hat{\rho}_{ij}(\hat{S}^{(i)}_*)| \leq c_1 s\sqrt{2\log(p)/n}$ for all $j \notin (\{i\} \cup \hat{S}^{(i)}_*)$.*

An explicit formula for c_1 can be worked out but is rather tedious; see the proofs of Theorems 2.2-2.3 for details. The first two claims of the theorem are still valid if $t_q^*(p,n) \asymp s\sqrt{2\log(p)/n}$ but $t_q^*(p,n) \leq c_1 s\sqrt{2\log(p)/n}$. In such a case, the difference is that, PCS may continue to run for finitely many steps (without immediate termination) after all signals are recruited.

Remark. We can slightly relax the condition (2.23) by allowing $\tau_p^* \sim r \cdot s\sqrt{\log(p)/n}$ for some constant $r > 0$. In this case, there exists a constant r^* that only depends on c_0 such that whenever $r > r^*$, we can find constants $\underline{c} = \underline{c}(c_0, r)$ and $\bar{c} = \bar{c}(c_0, r)$, so that Theorem 2.2 continues to hold when $\underline{c} \leq q \leq \bar{c}$. Furthermore, if we only want the first two claims of Theorem 2.2 to hold, we do not need the lower bound $\underline{c}$ for q.

Theorem 2.2 discusses the Screen step of the PCS for individual rows. The following theorem characterizes properties of the estimator $\hat{\Omega}^{pcs} = \hat{\Omega}^{pcs}(t_q^*, X; p, n)$, and is proved in the supplementary material [26].

Theorem 2.3 *Under conditions of Theorem 2.2, with probability at least $1 - o(p^{-2})$, each row of $\hat{\Omega}^{pcs}$ has the same support as the corresponding row of Ω, and $\|\hat{\Omega}^{pcs} - \Omega\|_{\max} \leq C\sqrt{\log(p)/n}$.*

While Theorem 2.3 is for $\|\hat{\Omega}^{pcs} - \Omega\|_{\max}$, the results can be extended to accommodate other types of matrix norms (e.g., $\|\hat{\Omega}^{pcs} - \Omega\|_1$).

Remark. In Theorems 2.1-2.2, we use the lower bound $(s-k)(\tau_p^*)^2$ for the EAL associated with signals that are At Large between the $(k-1)$-th and k-th Selecting Time. Such a bound is not tight, especially when a few smallest nonzero entries are much smaller than other nonzero entries (in magnitude). Here is a better bound. Suppose row i has s off-diagonal nonzeros, denoted as $\eta_1, \cdots, \eta_s$. We sort η_j^2 in the ascending order: $\eta_{(1)}^2 \leq \eta_{(2)}^2 \leq \ldots \leq \eta_{(s)}^2$. Then, the EAL is lower bounded by $\sum_{\ell=1}^{s-k} \eta_{(\ell)}^2$. Such a bound can help relax the condition (2.23) for Theorem 2.2, especially when our goal is not to show exact support recovery, but to control the number of signal nodes not recruited in the Screen step.

Remark. We control the stochastic fluctuations (2.22) by showing that for each $1 \leq i \leq p$ and $m \ll N$, with probability at least $1 - o(p^{-3})$,

$$\|\hat{\Sigma}^{W,W} - \Sigma^{W,W}\| \leq C\sqrt{m\log(p)/n}, \qquad \text{where} \quad W = \{i\} \cup \hat{S}_{m-1}^{(i)}. \tag{2.25}$$

If we replace $\hat{S}_{m-1}^{(i)}$ by a fixed subset S with $|S| = m-1$, then by basics in multivariate analysis, the factor $\sqrt{m}$ on the right hand side can be removed. In general, if we can find an upper bound for the number of possible realizations of $\hat{S}_{m-1}^{(i)}$, say, $K(p,m)$, then we can replace $\sqrt{m}$ by $\sqrt{\log(K(p,m))}$. In (2.25), $K(p,m) = \binom{p}{m}$ which is the most conservative bound. How to find a tighter bound for $K(p,m)$ is a difficult problem [1]. We conjecture that in a broad situation, a better bound is possible so (2.25) can be much improved.

At the same time, if we are willing to impose further conditions on Σ, then such a tighter bound is possible; we investigate this in Section 2.3.

2.3. *Consistency of PCS for much weaker signals.* In the above results, in order for PCS to be successful, we need $\tau_p^* \gg s\sqrt{\log(p)/n}$. We wish to relax this condition by considering

$$\tau_p^* \geq r \cdot \sqrt{2\log(p)/n}, \qquad \text{where } r > 0 \text{ is a fixed constant.} \tag{2.26}$$

We show PCS works in such cases if we put additional conditions on Σ. Let $\kappa^* = \kappa^*(\Sigma) = \max_{1\leq i\leq p} \kappa(i,\Sigma)$ and $\gamma^* = \gamma^*(\Sigma) = \min_{1\leq i\leq p} \gamma(i,\Sigma)$, where $\kappa(i;\Sigma) = \max_{j\notin\{i\}\cup S^{(i)}(\Omega)} \|(\Sigma^{S^{(i)}(\Omega),S^{(i)}(\Omega)})^{-1}\ \Sigma^{S^{(i)}(\Omega),\{j\}}\|_1$ and $\gamma(i;\Sigma) = \min_{j\notin(\{i\}\cup S^{(i)}(\Omega))} \{[\text{first diagonal of } (\Sigma^{\{j\}\cup S^{(i)}(\Omega),\{j\}\cup S^{(i)}(\Omega)})^{-1}]^{1/2}\}$; here, we always assume j as the first index listed in $\{j\}\cup S^{(i)}(\Omega)$. The quantity κ^* is motivated by a similar quantity in [44] for linear regressions, and γ^* is a normalizing factor which comes from the definition of partial correlations. Fix a constant $\delta \in (0,1)$. In this sub-section, we assume

$$\kappa^*(\Sigma)/\gamma^*(\Sigma) \leq 1-\delta, \qquad \Sigma(i,i) = 1, \qquad 1 \leq i \leq p; \tag{2.27}$$

the second assumption is only for simplicity in presentation. Introduce $\theta^* = \theta^*(\Sigma) = \min_{1\leq i\leq p} \theta(i,\Sigma)$, where $\theta(i,\Sigma) = \lambda_{\min}(\Sigma^{S^{(i)}(\Omega),S^{(i)}(\Omega)})$. The following theorem is proved in the supplemental material [26].

Theorem 2.4 *Fix $1 \leq i \leq p$ and apply the Screen step of PCS to row i. Suppose $\Sigma \in \mathcal{M}_p^*(s, c_0)$, (2.27) holds for some $\delta \in (0,1)$, $s^2\log(p) = o(n)$, the minimum signal strength τ_p^* satisfies (2.26) with $r \geq \sqrt{5\Omega(i,i)}[\theta^*(\Sigma)]^{-1}$ $\max\{\sqrt{\theta^*(\Sigma)} + 2\delta^{-1}, 2\sqrt{\Omega(i,i)}\}$, and the threshold $t_q^*(p,n)$ satisfies $\sqrt{5} < q \leq \theta^*(\Sigma)r/\Omega(i,i)$. With probability at least $1 - o(p^{-3})$,*

- *Before all signal nodes are recruited, PCS keeps running and recruits a signal node at each step.*
- *PCS takes exactly $|S^{(i)}(\Omega)|$ steps to terminate.*
- *When PCS stops, $|\hat{\rho}_{ij}(\hat{S}_*^{(i)})| \leq \sqrt{10\log(p)/n}$ for all $j \notin (\{i\}\cup\hat{S}_*^{(i)})$.*

By Theorem 2.4, the claim of Theorem 2.3 continues to hold, the proof of which is straightforward so we omit it. In Theorem, 2.4, we require $r \geq \sqrt{5\Omega(i,i)}[\theta^*(\Sigma)]^{-1}(\sqrt{\theta^*(\Sigma)} + 2\delta^{-1})$ and $r \geq 2\sqrt{5\Omega(i,i)}[\theta^*(\Sigma)]^{-1}\sqrt{\Omega(i,i)}\}$. The first condition ensures that PCS always recruits signal nodes before termination. The second one ensures the existence of a threshold by which PCS terminates immediately once all signals are recruited.

2.4. *Optimal classification phase diagram by HCT-PCS.* Come back to model (1.11) where $\tilde{X}_i \sim N(\mu^{\pm}, \Sigma)$, $\Omega = \Sigma^{-1}$, if $Y_i = \pm 1$, respectively. In this model, the optimality of HCT was justified carefully in [13, 22]. At the heart of the theoretical framework is the notion of *classification phase diagram*. Call the two-dimensional space calibrating the signal sparsity (fraction of nonzeros in the contrast mean vector $(\mu^+ - \mu^-)$) and signal strength (minimum magnitudes of the nonzero contrast mean entries) the *phase space*. The phase diagram is a partition of the phase space into three sub-regions, where successful classification is *relatively easy*, *possible but relatively hard*, and *impossible* simply because the signals are too rare and weak.

We say a trained classifier achieves the optimal phase diagram if it partitions the phase space in exactly the same way as the optimal classifier does. It was shown in [22, Theorm 1.1-1.3] that HCT achieves the optimal phase diagram (with some additional regularity conditions) provided that

- Ω is s_p-sparse, where $s_p \leq L_p$.
- Ω is known, or can be estimated by $\hat{\Omega}$ such that $\|\hat{\Omega} - \Omega\|_{\max} \leq L_p/\sqrt{n}$.

Here, $L_p > 0$ is a generic multi-log(p) term such that for any constant $c > 0$, $L_p p^{-c} \to 0$ and $L_p p^c \to \infty$.

We now consider HCT-PCS. By results in Sections 2.2-2.3, we have shown

$$\|\hat{\Omega}^{pcs} - \Omega\|_{\max} \leq C\sqrt{\log(p)/n}. \tag{2.28}$$

Therefore, HCT-PCS achieves the optimal phase diagrams in classification, provided that $s_p \leq L_p$. See [22] for details.

Note that the condition on s_p is relatively strict here. For much larger s_p (e.g., $s_p = p^{\vartheta}$ for some constant $0 < \vartheta < 1$), it remains unknown which procedures achieve the optimal phase diagram, even when Ω is known.

2.5. *Comparisons with other methods.* There are some existing theoretical results on exact support recovery of the precision matrix, including but are not limited to those on the glasso [34], CLIME [10], and scaled-lasso [37].

For exact support recovery, the glasso requires the so-called "Incoherent Conditions" (IC) [34]. The IC condition is relatively restrictive, which can be illustrated by the following simple example. Suppose p is divisible by 3, and Ω is block-wise diagonal where each diagonal block is a symmetric matrix $D \in \mathbb{R}^{3,3}$ satisfying $D(1,1) = D(2,2) = 1, D(1,2) = 0, D(1,3) = a, D(2,3) = b$, and $D(3,3) = c$, where $c^2 > a^2 + b^2$ so D is positive definite. In this example, the IC condition imposes a restriction $|a| + |b| + 2|ab| < 1$.

The conditions required for CLIME to achieve the exact support recovery is given in [10], which in our notations can be roughly translated into $\tau_p^* \geq$

$C\|\Omega\|_1/\sqrt{n}$, where τ_p^* is the minimum magnitude of the nonzero off-diagonals of Ω. In comparison, Theorem 2.4 says that if condition (2.27) holds, then PCS achieves exact support recovery provided that $\tau_p^* \geq C\sqrt{2\log(p)}/\sqrt{n}$. Note that the two sets of conditions overlap with each other, and there are many cases where the conditions for Theorem 2.4 hold but that for CLIME does not. For example, if Ω is a block-wise diagonal matrix, and within each block, all entries are nonzero. Then $\kappa^*(\Sigma)/\gamma^*(\Sigma) = 0$ and (2.27) is easily satisfied. However, it is possible that $\|\Omega\|_1 \gg \sqrt{2\log(p)}$, especially when some of the blocks are large. Therefore, in such a case, CLIME requires much larger τ_p^* to succeed than does PCS. As for scaled-lasso, note that the primary interest in [37] is on the convergence in terms of the matrix spectral norm, where conditions for exact support recovery are not given.

Note that the largest advantage of PCS is that, it is fast and allows for energy-efficient computing for very large matrices, and has nice results in real data analysis.

The method in [5] and FoBa [44, 45] are also related. However, the main results of [5] is on the case where Σ is sparse. Since the primary interest here is on the case where Ω is sparse, their results do not directly apply. The results in [44, 45] are on variable selection, and have not yet been adapted to precision matrix estimation. Recall that in Section 1.6, we have already carefully compared PCS with FoBa, from the perspective of real data applications: PCS is different from FoBa in philosophy, method and implementation, and yields much better classification results.

The results (numerical and theoretical) presented in this paper suggest that PCS is an interesting procedure and is worthy of future exploration. In particular, we believe that, with some technical advancements in proofs, the conditions required for the success of PCS can be largely weakened.

3. Simulations.

We conducted 4 different simulated experiments. The first one compares PCS with FoBa and the glasso in precision matrix estimation. The second one is similar, but the focus is on smaller p and we include CLIME for comparison. The third one investigates the robustness of PCS. The last one compares the classification behavior of HCT-PCS with those of HCT-Foba, HCT-glasso, nHCT, SVM, and RF.

3.1. *Experiment 1 (precision matrix estimation).*

Experiment 1 consists of three sub-experiments, 1a–1c. In each sub-experiment, we generate samples $X_i \overset{iid}{\sim} (0, \Omega^{-1})$, $i = 1, 2, \ldots, n$, where $\Omega \in \mathbb{R}^{p,p}$, for 10 repetitions. For any $\hat{\Omega}$, an estimate of Ω, we measure the performance by the average errors across 10 different repetitions. We use four different error measures: spectrum norm, Frobenius norm, the matrix ℓ^1-norm of $(\hat{\Omega} - \Omega)$, and the matrix

Hamming distance between $\hat{\Omega}$ and Ω. The first three error measures are as in textbooks, the last one is defined by

$$\text{Hamm}_p(\hat{\Omega}, \Omega) = \frac{1}{p} \sum_{1 \leq i,j \leq p} 1\{\text{sgn}(|\hat{\Omega}(i,j)|) \neq \text{sgn}(|\Omega(i,j)|)\}, \tag{3.29}$$

where $\text{sgn}(x) = 1$ if $x > 0$ and $\text{sgn}(x) = 0$ if $x = 0$. Alternatively, we can replace the factor p^{-1} by 1 or p^{-2}, but the resultant values would be either too large or too small; the current one is the best for presentations.

In these experiments, matrix singularity is not as extreme as in the microarray data, so we use PCS and FoBa without the ridge regularization.

For PCS, we take the tuning parameter L to be 15 in experiments 1a–1b for the algorithm generally stops after 10 steps due to the simple structure of Ω. In experiment 1c, we use $L = 30$ because Ω is more complex. For tuning parameter q, we test q from 0.5 to 3 with increment of 0.5 in experiment 1a and 1b. We find that for $q \leq 0.5$ or $q \geq 2.5$, the errors are higher, while the errors remain similar for $1 \leq q \leq 2$, so we use $q = 1.5$. In experiment 1c, we test q from 0.25 to 2 with increment of 0.25. We find that for $q \leq 0.25$ or $q \geq 1.5$, the errors are higher than $0.5 \leq q \leq 1.25$, so we use $q = 0.75$. For FoBa, we set L the same as in PCS.

For the glasso, we set the tuning parameter λ as 0.5. In principle, the smaller the λ we use, the slower the algorithm, but more accurate the estimate. For our experiments, it takes about 10 hours for experiments $1a$-$1b$ and more than 24 hours for experiment $1c$ with $\lambda = 0.5$, so we do not consider λ smaller than 0.5, for it may take substantially longer. On the other hand, we should choose λ as small as possible, for the sake of accuracy. For these reasons, we take $\lambda = 0.5$. Of course, if p is relatively small (i.e., only a few hundreds), then we can use a more careful choice of λ; see Section 3.2.

We now describe Ω in three sub-experiments. For experiments 1a and 1c, we set $(p, n) = (5000, 1000), (2000, 1000), (1000, 500)$. For experiment 1b, we set $(p, n) = (4500, 1000), (3000, 1000), (1500, 500)$ so that p is divisible by 3.

Experiment 1a: $\Omega(i,j) = 1\{i = j\} + \rho \cdot 1\{|i-j| = 1\}$, $\rho = 0.4$, $1 \leq i, j \leq p$. Here, the IC condition (see Section 2.5) for the glasso holds, but no longer holds if we increase ρ slightly.

Experiment 1b: Ω is a block-wise diagonal matrix, and each diagonal block is a 3×3 symmetric matrix A satisfying $A(1,1) = A(2,2) = A(3,3) = 1$, $A(1,2) = 0$, $A(1,3) = 0.5$, and $A(2,3) = 0.7$. This matrix Ω is positive definite but does not satisfy the IC condition.

Experiment 1c: We generate Ω as follows. First, we generate a $p \times p$ Wigner matrix W [38] (the symmetric matrix with 0 on all the diagonals and iid Bernoulli(ϵ) random variables for entries on the upper triangle; here we take

$\epsilon = .01$). Next, we let $\Omega^* = .5W + \vartheta I_p$, where I_p is the $p \times p$ identity matrix and $\vartheta = \vartheta(W)$ is such that the conditional number of Ω^* (the ratio of the maximal and the minimal singular values) is p. Last, we scale Ω^* to have unit diagonals and let Ω be the resultant matrix.

TABLE 4

Estimation errors (with standard deviations in brackets) for Experiment 1a.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
5000	1000	0.27(0.021)	1.19(0.003)	0.78(0.014)	0.34(0.033)	1.23(0.003)	2.45(0.070)
2000	1000	0.26(0.027)	1.18(0.003)	0.70(0.018)	0.34(0.035)	1.23(0.005)	2.18(0.107)
1000	500	0.34(0.033)	1.19(0.003)	1.14(0.025)	0.45(0.051)	1.24(0.004)	3.24(0.174)
		Frobenius norm			Matrix Hamming distance		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
5000	1000	4.39(0.057)	49.00(0.013)	23.08(0.067)	0.00(0.000)	0.00(0.001)	24.93(0.021)
2000	1000	2.79(0.036)	30.99(0.012)	13.03(0.037)	0.00(0.000)	0.00(0.002)	24.43(0.030)
1000	500	2.83(0.099)	21.91(0.019)	14.89(0.094)	0.00(0.000)	0.04(0.011)	24.12(0.038)

TABLE 5

Estimation errors (with standard deviations in brackets) for Experiment 1b.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
4500	1000	0.30(0.022)	1.23(0.004)	0.73(0.017)	0.36(0.035)	1.49(0.005)	2.13(0.043)
3000	1000	0.29(0.027)	1.22(0.004)	0.70(0.018)	0.35(0.039)	1.49(0.006)	2.01(0.047)
1500	500	0.36(0.018)	1.23(0.003)	1.15(0.022)	0.43(0.028)	1.51(0.006)	3.22(0.239)
		Frobenius norm			Matrix Hamming distance		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
4500	1000	3.99(0.059)	49.83(0.009)	19.47(0.069)	0.00(0.000)	0.69(0.002)	26.90(0.019)
3000	1000	3.25(0.029)	40.69(0.008)	14.96(0.059)	0.00(0.000)	0.68(0.003)	26.78(0.019)
1500	500	3.30(0.092)	28.76(0.013)	17.95(0.118)	0.00(0.000)	1.47(0.044)	26.58(0.035)

TABLE 6

Estimation errors (with standard deviations in brackets) for Experiment 1c.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
5000	1000	2.56(0.009)	4.13(0.002)	4.47(0.000)	5.05(0.149)	13.50(0.675)	6.29(0.056)
2000	1000	0.53(0.009)	2.79(0.001)	3.11(0.000)	1.84(0.097)	7.25(0.221)	5.42(0.013)
1000	500	1.00(0.068)	2.13(0.001)	2.41(0.001)	3.25(0.305)	13.27(0.249)	4.11(0.009)
		Frobenius norm			Matrix Hamming distance		
p	n	PCS	glasso	FoBa	PCS	glasso	FoBa
5000	1000	35.46(0.068)	55.41(0.025)	72.18(0.004)	35.22(0.103)	69.68(0.668)	55.80(0.037)
2000	1000	10.61(0.054)	33.88(0.020)	43.63(0.003)	5.89(0.067)	34.60(0.295)	25.94(0.077)
1000	500	10.65(0.074)	22.34(0.013)	29.23(0.003)	8.29(0.085)	14.70(0.093)	15.27(0.132)

The results for three experiments are summarized in Tables 4, 5, and 6, correspondingly, in terms of four error measures aforementioned. For experiments 1a–1b, it suggests that (a) PCS outperforms the glasso and FoBa in all four different error measures, especially in Hamming loss, where PCS has no errors in all cases (and thus exact support recovery of Ω); (b) the glasso and FoBa have similar performance in terms of the ℓ^1-norm and Hamming distance, but the glasso is significantly inferior to FoBa in terms of the spectral norm and Frobenius norm. For experiment 1c, Table 6 shows that the glasso is not that competitive to FoBa as in the previous two experiments, while PCS still has a dominant advantage over the glasso and FoBa when both p and n get larger, especially in terms of the Hamming distance.

3.2. *Experiment 2 (PCS for relatively small p).* In this experiment, we wish to include CLIME for comparison. Since CLIME is computationally slow for large p, we choose moderately large p. The experiment consists of 3 sub-experiments, 2a-2c, where we use the same settings as in Experiment 1a-1c, respectively, except for that $(p, n) = (200, 100)$. For PCS, FoBa, the glasso, and CLIME, the tuning parameters are set as follows. For FoBa, we set $L = 15$. For PCS, we also set $L = 15$. Additionally, we find that when the tuning parameter q range from .1 to 2 (with an increment of .1), the error rates are relatively flat for $0.8 < q < 1.6$ (and higher outside this range), so we set $q = 1$ for convenience. For glasso and each of the 4 error measures we consider, we pick the λ in the set $\{.05, .1, .15, \ldots, 1\}$ that has the smallest error rates (i.e., the "ideal" tuning parameter). For CLIME and each of the four error measures, we pick the "ideal" tuning parameter from the set $\{.1, .2, \ldots, 1\}$ for Experiment 2a-2b and the "ideal" tuning parameter from the set $\{.01, .02, .03, \ldots, .2\}$ for Experiment 2c (the interesting range for λ in Experiment 2c is different from that in Experiment 2a-2b).

The results for experiment 2a-2c are summarized in Tables 7, 8, and 9, respectively. The results show that PCS is generally competitive. Also, the behavior of CLIME is comparable to PCS in three choices of the error measures, but is much less satisfactory in the Hamming loss.

3.3. *Experiment* 3 *(robustness).* We have 3 sub-experiments, 3a-3c, where we use the same settings as in experiment 1a-1c, respectively. However, for each experiment, instead of applying PCS to all samples, we apply PCS to 90% of samples that are randomly selected. For each of the 4 error measures we consider, we repeat the subsampling for 10 times independently and compute the average errors. The results are in Table 10, 11 and 12, respectively, where the columns of $\hat{\Omega}_{\text{sample}} - \Omega$ correspond to the average error measures for 10 different sub-sampling. We observe that using 90% of the samples yield almost the same results as that of using all samples, especially in terms of the Hamming loss. This suggest that PCS is reasonably robust.

3.4. *Experiment 4 (classification).* In this experiment, we take Ω to be the tri-diagonal matrix as in experiment 1a, calibrated by the parameter ρ.

TABLE 7
Estimation errors (with standard deviations in brackets) for Experiment 2a.

		Spectrum norm				Matrix ℓ^1-norm			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	1.19(0.200)	0.85(0.017)	5.46(0.276)	1.07(0.032)	1.79(0.373)	1.31(0.034)	13.12(0.657)	1.27(0.055)
		Frobenius norm				Matrix Hamming distance			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	4.72(0.243)	5.51(0.104)	30.29(1.040)	7.77(0.137)	0.39(0.044)	0.46(0.064)	22.61(0.160)	197.01(0.000)

TABLE 8
Estimation errors (with standard deviations in brackets) for Experiment 2b.

		Spectrum norm				Matrix ℓ^1-norm			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	0.98(0.320)	1.28(0.009)	5.11(0.484)	1.18(0.027)	1.30(0.454)	1.83(0.016)	11.63(1.205)	1.52(0.036)
		Frobenius norm				Matrix Hamming distance			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	3.00(0.253)	12.76(0.024)	27.77(0.549)	5.82(0.246)	0.04(0.021)	15.75(0.471)	24.45(0.176)	197.7(0.000)

TABLE 9
Estimation errors (with standard deviations in brackets) for Experiment 2c.

		Spectrum norm				Matrix ℓ^1-norm			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	1.30(0.106)	1.34(0.032)	1.72(0.008)	1.30(0.023)	2.12(0.212)	3.02(0.039)	2.98(0.065)	3.18(0.074)
		Frobenius norm				Matrix Hamming distance			
p	n	PCS	glasso	FoBa	CLIME	PCS	glasso	FoBa	CLIME
200	100	6.07(0.232)	7.82(0.025)	10.90(0.015)	7.74(0.086)	1.26(0.084)	3.496(0.074)	8.93(0.288)	196.9(0.000)

Also, following [22], we consider the most challenging "rare and weak" setting where the contrast mean vector μ only has a small fraction of nonzeros and the nonzeros are individually small. In detail, let ν_a be the point mass at a. For two numbers (ϵ_p, τ_p) that may depend on p, we generate the scaled vector $\sqrt{n}\mu$ from the mixture of two point masses: $\sqrt{n}\mu(j) \overset{iid}{\sim} (1-\epsilon_p)\nu_0 + \epsilon_p \nu_{\tau_p}$.

In this experiment, we take $(p, n, \rho, \epsilon_p, \tau_p) = (5000, 1000, 0.4, 0.1, 3.5)$. For (μ, Ω) generated as above, the simulation contains the following main steps:

1. Generate n samples $(\tilde{X}_i, Y_i)$, $1 \leq i \leq n$, by letting $Y_i = 1$ for $i \leq n/2$ and $Y_i = -1$ for $i > n/2$, and $\tilde{X}_i \sim N(Y_i \cdot \mu, \Omega^{-1})$.
2. Split the n samples into training and test sets by following exactly the same procedure in Section 1.4. The only difference is that we use 10 data splittings and 10 cv-splittings here.
3. Use the training set to build all classifiers (HCT-PCS, HCT-FoBa, HCT-glasso, nHCT, SVM and RF), apply them to the test set, and then record the test errors.

The results are summarized in Table 13 in terms of both the average error across 10 data splittings and the minimum error in 10 data splittings. It suggests that HCT-PCS outperforms other HC-based classifiers; in particular, HCT-PCS significantly outperforms nHCT and HCT-glasso. In addition, both SVM and RF are less competitive compared to HCT-PCS. This is con-

TABLE 10
Robustness test errors (with standard deviations in brackets) for Experiment 3a.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
5000	1000	0.27	0.29(0.018)	0.09(0.006)	0.37	0.38(0.026)	0.12(0.012)
2000	1000	0.27	0.28(0.022)	0.08(0.010)	0.37	0.38(0.030)	0.11(0.016)
1000	500	0.35	0.39(0.023)	0.20(0.010)	0.45	0.48(0.029)	0.25(0.014)
		Frobenius norm			Matrix Hamming distance		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
5000	1000	4.49	4.72(0.020)	1.46(0.017)	0.00	0.00(0.000)	0.00(0.000)
2000	1000	2.84	2.99(0.019)	0.92(0.015)	0.00	0.00(0.000)	0.00(0.000)
1000	500	2.91	3.09(0.025)	0.99(0.027)	0.00	0.00(0.000)	0.00(0.000)

TABLE 11
Robustness test errors (with standard deviations in brackets) for Experiment 3b.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
5000	1000	0.31	0.31(0.015)	0.11(0.014)	0.35	0.38(0.024)	0.13(0.014)
2000	1000	0.29	0.30(0.016)	0.10(0.005)	0.37	0.38(0.027)	0.12(0.007)
1000	500	0.39	0.42(0.018)	0.14(0.019)	0.45	0.51(0.038)	0.17(0.015)
		Frobenius norm			Matrix Hamming distance		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
4500	1000	4.03	4.25(0.023)	1.34(0.012)	0.00	0.00(0.000)	0.00(0.000)
3000	1000	2.84	3.46(0.025)	1.08(0.024)	0.00	0.00(0.000)	0.00(0.000)
1500	500	3.38	3.56(0.032)	1.11(0.032)	0.00	0.00(0.000)	0.00(0.000)

sistent with the theoretical results in [22], where it was shown that given a sufficiently accurate estimate of Ω, the HCT classifier has the optimal classification behavior in the "rare and weak" settings associated with the sparse Gaussian graphical model (1.11).

4. Discussions and extensions.

This paper is closely related to areas such as precision matrix estimation, classification, variable selection, and inference on "rare and weak" signals, and has many possible directions for extensions. Below, we mention some of such possibilities.

TABLE 12
Robustness test errors (with standard deviations in brackets) for Experiment 3c.

		Spectrum norm			Matrix ℓ^1-norm		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
5000	1000	2.56	2.63(0.004)	0.57(0.038)	5.39	5.38(0.179)	2.15(0.090)
2000	1000	0.54	0.60(0.011)	0.41(0.011)	1.97	2.13(0.135)	1.47(0.050)
1000	500	0.99	1.14(0.064)	0.81(0.080)	3.44	3.81(0.292)	3.26(0.376)
		Frobenius norm			Matrix Hamming distance		
p	n	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$	$\hat{\Omega}-\Omega$	$\hat{\Omega}_{\text{sample}}-\Omega$	$\hat{\Omega}_{\text{sample}}-\hat{\Omega}$
5000	1000	35.50	36.27(0.031)	16.83(0.070)	35.34	37.88(0.043)	14.16(0.089)
2000	1000	10.55	11.98(0.046)	7.71(0.041)	5.85	7.23(0.086)	6.70(0.088)
1000	500	10.70	12.02(0.146)	8.69(0.116)	8.19	9.30(0.179)	9.05(0.156)

The precision matrix can either be the direct quantity of interest (e.g., genetic regulatory networks), or a quantity that can be used to improve the results of inferences. Examples include classical methods of Hotelling's χ^2-test, discriminant analysis [13, 20], and the recent work on Innovated Higher Criticism [24]. In these examples, a good estimate of the precision matrix could largely improve the results of the inferences.

The theoretical results in the paper can be extended in various directions. For example, in this paper, we assume Ω is strictly sparse in the sense that in each row, most of the entries are exactly 0. Such an assumption can be largely relaxed. Also, the theoretical results presented in this paper focus on when it is possible to obtain exact support recovery. The results are extendable to the cases where we wish to measure the loss by matrix spectral norm or matrix ℓ^2-norm. In particular, we mention that if the ultimate goal is for classification, it is not necessary to fully recover the support of the precision matrix. A more interesting problem (but more difficult) is to study how the estimation errors in the precision matrix affect the classification results.

PCS needs a threshold tuning parameter q (it also uses the ridge parameter δ and a maximal step size parameter L, which we usually set by $(\delta, L) = (.1, 30)$; PCS is relatively insensitive to the choices of (δ, L)). When we use PCS for classification, we determine q by cross validation, which increases the computation costs by many times. The same drawback applies to other classifiers, such as HCT-FoBa, HCT-glasso, SVM, and RF.

The choice of $L = 30$ is not necessarily the best. We have tested PCS with $L = 10, 20, 30, 40, 50, 60$ and $L = 10, 20, 30, 40, 50, 60, 70, 80, 90$ for the rats data and the liver data, respectively. The average error rates of 25 different data splittings are shown in Figure 7. The results suggest that using a larger L (at least in the range we consider) would give better errors, but not significantly (the computational time, however, is longer, by a factor). In principle, we could also use cross validation to select the best L. However, this would increase the computation time by a few times at least, without much gain in the error rates. It is of interest to find an approach where we can select L in a more efficient way; we leave this to the future work.

From both a theoretical and practical perspective, we wish to have a

TABLE 13
Comparison of classification error rates for Experiment 4 (based on 10 independent data-splitting; 11.08 *means* 11.08%*)*

	HCT-PCS	HCT-Foba	HCT-glasso	nHCT	SVM	RF
average error	11.08	12.11	43.67	32.02	20.03	35.51
'best' error	8.73	10.54	37.95	21.99	18.98	31.93

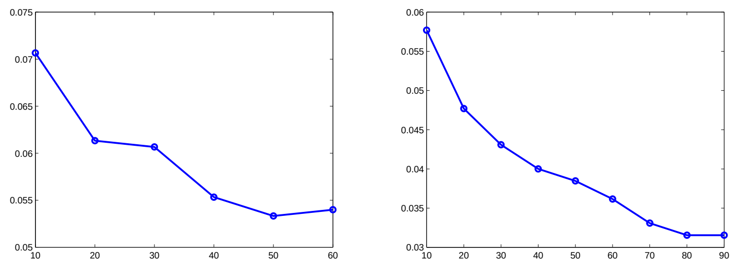

FIG 7. *Left: mean error rates of 25 different data splittings for rats data using different L. Right: mean error rates of 25 different data splittings for liver data using different L.*

trained classifier that is tuning free. In Donoho and Jin (2008) [13], we propose HCT as a tuning free classifier that enjoys optimality, but unfortunately the method is only applicable to the case where Ω is known. How to develop a tuning free optimal classifier for the case where Ω is unknown is a very interesting problem. For reasons of space, we leave to future work.

Intellectually, our work on HCT classification is closely related to [18, 19, 27], but is different in important ways, especially on the part of data-driven threshold choice and on phase diagrams. The work is also closely related to other development on Higher Criticism. See for example [3, 14, 29, 47, 28].

In Section 2.4, we show that HCT achieves the optimal phase diagram given a relatively stringent sparsity constraint (for exact support recovery, however, only a much less stringent constraint is required). In the much less sparse cases, it is unclear how to achieve the optimal phase diagram, even in the ideal case where Ω is known [22]. Still, PCS may largely help in improving the classification results of HCT in such a case. See Table 1 for example, where HCT-PCS yields satisfactory classification behaviors, even when we don't believe the underlying Ω satisfies the very stringent sparsity conditions we specified in Section 2.4.

Our philosophy in classification is very different from that in [6, 10], where they suggest that bypassing the estimate of the precision matrices may give better classification results. Their point is valid for cases where we don't have a good estimate for the precision matrix. However, as research in this area progresses, more and more faster algorithms and better estimates of the precision matrices become available, and it is desirable to utilize the estimates of the precision matrices for better classification results. Our results strongly support our philosophy: in Table 1, the classification error rates of HCT-PCS is 40% lower than that of naive HCT (nHCT); in nHCT, we pretend the precision matrix is diagonal, and implement HCT without any estimate of the off-diagnals of the precision matrix.

In the two-class classification settings we consider (e.g., (1.11)), we assume the two classes share a common covariance matrice, and we are mainly using the contrast mean to differentiate two classes. For the two data sets we consider, the error rates of HCT-PCS are relatively low (e.g., 5.7% and 4.2%), which suggests that the assumption of equal covariance matrices is reasonable. At the same time, we must note that it is preferable to use a model which allows for unequal covariance matrices. Take gene microarrays for example, it is usually believed that some genes would change their functions individually or cooperatively between the two classes. In such settings, we can use both the contrast mean and the difference between two covariances matrices to differentiate two classes. But in order to do so, we need (a) to estimate two precision matrices separately, each using part of the samples, (b) more tuning parameters and more computing time, (c) to develop a more sophisticated classifier. How to deal with these issues is an interesting problem, which we leave for the future study.

The idea in this paper can also be used to analyze the SNP data. Modern SNP data sets may have many more features (e.g., $p = 250K$) than a typical microarray data set. While the sheer large size poses great challenges for computation, we must note that in many of such studies on SNP, the (population) covariance matrix among different SNPs is banded. Such a nice feature can help to substantially reduce the computational burden.

5. Proofs.

We show Lemmas 2.1-2.2 which include the key idea of PCS. Other theorems and lemmas are proved in the supplementary material [26].

5.1. *Proof of Lemma 2.1.* Fix i and write for short $S_0 = S^{(i)}(\Omega)$ and ω' as the i-th row of Ω. For each m, we define $U_m = \{i, j_1, \cdots, j_m\}$. Then

(5.30)
$$\rho_{ij_m}(S^{(i)}_{m-1}) = \frac{-1 \cdot [\text{first row last column of } (\Sigma^{U_m,U_m})^{-1}]}{[\text{product of the first and last diagonals of } (\Sigma^{U_m,U_m})^{-1}]^{1/2}}.$$

We need some notations to simplify the matrix $(\Sigma^{U_m,U_m})^{-1}$. Fix $k \geq 1$. Introduce the set $V = \{i, j_1, \cdots, j_{m(k)}\} \cup (S_0 \backslash S^{(i)}_{m(k)})$, where i is the first index and j_m is the $(m+1)$-th index in the set, $1 \leq m \leq m(k)$. Let $A = \Sigma^{V,V}$. For each $m \geq 1$, we partition A into blocks corresponding to the first $(m+1)$-th indices and the remaining ones

$$A = \begin{pmatrix} A_{11}^{(m)} & A_{12}^{(m)} \\ A_{21}^{(m)} & A_{22}^{(m)} \end{pmatrix},$$

so that $\Sigma^{U_m,U_m} = A_{11}^{(m)}$. For notation simplicity, we shall omit all the superscripts and write $A_{11}^{(m)}$ as A_{11}. Using the matrix inverse formula,

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} + B_{12}B_{22}^{-1}B_{21} & -B_{12}B_{22}^{-1} \\ -B_{22}^{-1}B_{21} & B_{22}^{-1} \end{pmatrix}, \tag{5.31}$$

where $B_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$, $B_{12} = A_{11}^{-1}A_{12}$ and $B_{21} = A_{21}A_{11}^{-1}$.

Now, we show the claim. Since $V \supset S_0$, Lemma 1.1 implies that the first row of A^{-1} is equal to ω' restricted to V. Combining this with (5.31), for each $m(k-1) < m < m(k)$,

$$\text{first row last column of } [A_{11}^{-1} + B_{12}B_{22}^{-1}B_{21}] = \omega(j_m) = 0,$$
$$-1 \cdot \text{first row of } B_{12}B_{22}^{-1} = (\omega^{V\setminus U_m})'.$$

Also, by definition, $B_{21} = A_{21}A_{11}^{-1}$. Combining the above,

$$\text{first row last column of } A_{11}^{-1} = (\omega^{V\setminus U_m})'A_{21} \cdot \text{last column of } A_{11}^{-1}. \tag{5.32}$$

To simplify (5.32), we introduce a vector $\eta_k \in \mathbb{R}^{|V|}$ such that $\eta_k(j+1) = 0$ for $0 \leq j < m(k)$ and $\eta_k(j+1) = (\omega^V)(j+1)$ for $m(k) \leq j \leq |V|-1$. For each $m(k-1) < m < m(k)$, let $J_m = \{1, \cdots, m+1\}$ and $J_m^c = \{m+2, \cdots, |V|\}$. Noting $\omega^{V\setminus U_m} = (\omega^V)^{J_m^c} = (\eta_k)^{J_m^c}$, $A_{21} = A^{J_m^c, J_m}$ and $(\eta_k)^{J_m} = 0$, we have

$$(\omega^{V\setminus U_m})'A_{21} = (\eta_k'A)^{J_m}. \tag{5.33}$$

Moreover, let $A = LL'$ be the Cholesky decomposition of A, where L is a lower triangular matrix with positive diagonals. By basics of Cholesky decomposition, for $L_m = L^{J_m,J_m}$, $A_{11} = A^{J_m,J_m} = L_mL_m'$ is the Cholesky decomposition of A_{11}, and L_m satisfies $L_m^{-1} = (L^{-1})^{J_m,J_m}$. Therefore,

$$A_{11}^{-1} = [(L^{-1})']^{J_m,J_m}(L^{-1})^{J_m,J_m}. \tag{5.34}$$

The nice thing about (5.33)-(5.34) is that on the right hand sides, (η_k, A, L) only depend on k but not m (while on the left hand sides, A_{21} and A_{11} depend on m). This allows us to stack the expressions for different m.

Plugging (5.33)-(5.34) into (5.32) and noting that L^{-1} is lower triangular,

$$\text{first row last column of } A_{11}^{-1} = [\eta_k'A(L^{-1})']^{J_m} \cdot \text{last column of } (L^{-1})^{J_m,J_m}$$
$$= L^{-1}(m+1, m+1) \cdot q(m+1), \qquad q \equiv L^{-1}A'\eta_k.$$

This gives the numerator of (5.30). For the denominator, by (5.31) and (5.34), (a) the first diagonal of $A_{11}^{-1} \geq A^{-1}(1,1) \geq \lambda_{\max}^{-1}(A)$, and (b) the last diagonal of $A_{11}^{-1} = [L^{-1}(m+1, m+1)]^2$. Combining the above with (5.30),

$$\rho_{ij_m}^2(S_{m-1}^{(i)}) \leq \lambda_{\max}(A) \cdot q^2(m+1). \tag{5.35}$$

Since (5.35) holds for each $m(k-1) < m < m(k)$, we stack the results for all m and obtain $\sum_{m(k-1)<m<m(k)} \rho^2_{ij_m}(S^{(i)}_{m-1}) \leq \lambda_{\max}(A) \cdot \|q\|^2 \leq \lambda^2_{\max}(A) \cdot \|\eta_k\|^2$. Here, the last inequality is due to $A = L'L$ and $q = L^{-1}A'\eta = L'\eta$. The claim then follows by noting that $\|\eta_k\|^2 = \sum_{j\in(S_0\setminus S^{(i)}_{m(k)-1})} \omega^2(j_m)$ and that A is a principal submatrix of Σ with size $\leq m(k)+s-k$. □

5.2. *Proof of Lemma 2.2.* Fixing $1 \leq i \leq p$ and $m \geq 1$, we adopt the notations S_0, ω' and U_m as in the proof of Lemma 2.1. Let $W = S_0 \setminus S^{(i)}_{m-1}$. For each $j \in W$, let $V_j = \{i, j_1, \cdots, j_{m-1}, j\}$ and suppose i and j are the first and last indices in the set, respectively. By definition,

$$\rho_{ij}(S^{(i)}_{m-1}) = \frac{-1 \cdot [\text{first row last column of } (\Sigma^{V_j,V_j})^{-1}]}{[\text{product of first and last diagonals of } (\Sigma^{V_j,V_j})^{-1}]^{1/2}}. \tag{5.36}$$

Write $\Sigma_m = \Sigma^{U_{m-1},U_{m-1}}$ and $\eta_j = \Sigma^{U_{m-1},\{j\}}$ for short. By basic algebra,

$$(\Sigma^{V_j,V_j})^{-1} = \begin{pmatrix} \Sigma_m^{-1} + A & -[\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1}\eta_j'\Sigma_m^{-1} \\ -[\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1}\Sigma_m^{-1}\eta_j & [\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1} \end{pmatrix},$$

where A is a positive-definite matrix. It follows that

$$\begin{aligned} &\text{first row last column of } (\Sigma^{V_j,V_j})^{-1} = -1 \cdot [\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1} e_j'\Sigma_m^{-1}\eta_j, \\ &\text{last diagonal of } (\Sigma^{V_j,V_j})^{-1} = [\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1}, \end{aligned}$$

where $e_1 = (1, 0, \cdots, 0)'$. As a result,

$$\sum_{j\in W} \rho^2_{ij}(S^{(i)}_{m-1}) = \sum_{j\in W} \frac{[\Sigma(j,j) - \eta_j'\Sigma_m^{-1}\eta_j]^{-1}}{(\Sigma^{V_j,V_j})^{-1}(1,1)} (e_1'\Sigma_m^{-1}\eta_j)^2. \tag{5.37}$$

Let $D = \Sigma^{W,W} - \Sigma^{W,U_m}\Sigma_m^{-1}\Sigma^{U_m,W}$ and $H = \Sigma_m^{-1}\Sigma^{U_m,W}$. Then

$$\sum_{j\in W} \rho^2_{ij}(S^{(i)}_{m-1}) \geq \frac{1}{\max_{j\in W}(\Sigma^{V_j,V_j})^{-1}(1,1)} \|e_1'H[\text{diag}(D)]^{-1/2}\|^2. \tag{5.38}$$

Below, we make a connection between ω and the right hand side of (5.38). Introduce set $V = \{i, j_1, \cdots, j_{m-1}\} \cup W$ such that i is the first index and j_k is the $(k+1)$-th index, $1 \leq k \leq m-1$. Using the matrix inverse formula,

$$(\Sigma^{V,V})^{-1} = \begin{pmatrix} \Sigma_m & \Sigma^{U_{m-1},W} \\ \Sigma^{W,U_{m-1}} & \Sigma^{W,W} \end{pmatrix}^{-1} = \begin{pmatrix} \Sigma_m^{-1} + HD^{-1}H & -HD^{-1} \\ -D^{-1}H' & D^{-1} \end{pmatrix}.$$

Since $V \supset S_0$, by Lemma 1.1, the first row of $(\Sigma^{V,V})^{-1}$ coincides with $(\omega^V)'$. In particular,

$$e_1' H D^{-1} = -(\omega^W)'. \tag{5.39}$$

Furthermore, since $V_j \subset V$,

$$(\Sigma^{V_j,V_j})^{-1}(1,1) \leq (\Sigma^{V,V})^{-1}(1,1) \leq \lambda_{\min}^{-1}(\Sigma^{V,V}). \tag{5.40}$$

Plugging (5.39)-(5.40) into (5.38) gives that $\sum_{j\in W} \rho_{ij}^2(S_{m-1}^{(i)}) \geq \lambda_{\min}(\Sigma^{V,V}) \cdot \|[\mathrm{diag}(D)]^{-1/2} D\omega^W\|^2$. Note that $\|[\mathrm{diag}(D)]^{-1/2} D\omega^W\|^2 \geq \lambda_{\max}^{-1}(D)\lambda_{\min}^2(D) \cdot \|\omega^W\|^2$. Moreover, the eigenvalues of D are between $\lambda_{\min}(\Sigma^{V,V})$ and $\lambda_{\max}(\Sigma^{V,V})$. Therefore, $\sum_{j\in W} \rho_{ij}^2(S_{m-1}^{(i)}) \geq \frac{\lambda_{\min}^3(\Sigma^{V,V})}{\lambda_{\max}(\Sigma^{V,V})} \cdot \|\omega^W\|^2$. The claim follows by noting that $W = S_0 \setminus S_{m-1}^{(i)}$ and that the size of $\Sigma^{V,V}$ is $|W| + m \leq m + s$. $\square$

SUPPLEMENTARY MATERIAL

Supplementary Material for "Partial Correlation Screening for Estimating Large Precision Matrices, with Applications to Classification"
(doi: COMPLETED BY THE TYPESETTER; .pdf). Owing to space constraints, some technical proofs are relegated to a supplementary document.

References.

[1] An, H., Huang, D., Yao, Q. and Zhang, C. H. (2014). Stepwise searching for feature variables in high-dimensional linear regression. *Manuscript.*

[2] Anonymous, (2006 (retrieved)). Elephant and the blind men. *Jain Stories.*

[3] Arias-Castro, E., Candes, E. and Plan, Y. (2011). Global testing under sparse alternatives: ANOVA, multiple comparisons and the higher criticism. *Ann. Statist.* **39** 2533-2556.

[4] Bi, J., Bennett, K., Embrechts, M., Breneman, C. and Song, M. (2003). Dimensionality reduction via sparse support vector machines. *J. Mach. Learn. Res.* **3** 1229–1243.

[5] Bickel, P. and E., L. (2008). Regularized estimation of large covariance matrices. *Ann. Statist.* **36** 199–227.

[6] Bickel, P. J. and Levina, E. (2004). Some theory for Fisher's linear discriminant function,'naive Bayes', and some alternatives when there are many more variables than observations. *Bernoulli* 989–1010.

[7] Breiman, L. (2001). Random forests. *Mach. Learn.* **24** 5–32.

[8] Buhlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Springer.

[9] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **2** 121–167.

[10] Cai, T., Liu, W. and Luo, X. (2011). A constrained l_1 minimization approach to sparse precision matrix estimation. *J. Amer. Statist. Assoc.* **106** 594–607.

[11] Cawley, G. and Talbot, N. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.* **11** 2079–2107.

[12] Dettling, M. and Buhlmann, P. (2003). Boosting for tumor classification with gene expression data. *Bioinformatics* **19** 1061–1069.

[13] Donoho, D. and Jin, J. (2008). Higher Criticism Thresholding: Optimal feature selection when useful features are rare and weak. *Proc. Natl. Acad. Sci.* **105** 14790–14795.

[14] Donoho, D. and Jin, J. (2015). Higher Criticism for large-scale inference: especially for rare and weak effects. *Statistical Science* **30** 1–25.

[15] Donoho, D., Tsaig, Y., Drori, I. and Starck, J.-L. (2012). Sparse solution of undetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **58** 1094-1121.

[16] Efron, B. (2004). Large-scale simultaneous hypothesis testing: the choice of a null hypothesis. *J. Amer. Statist. Assoc.* **99** 96-104.

[17] Efron, B. (2009). Empirical Bayes estimates for large-scale prediction problems. *J. Amer. Statist. Assoc.* **104** 1015–1028.

[18] Fan, J. and Fan, Y. (2008). High-dimensional classification using features annealed independent rules. *Ann. Statist.* **36** 2605-2637.

[19] Fan, J., Feng, Y. and Tong, X. (2012). A road to classification in high dimension space: the regularized optimal affine discriminant. *J. Roy. Statist. Soc.* **74** 745-771.

[20] Fan, J., Ke, Z. T., Liu, H. and Xia, L. (2015). QUADRO: A supervised dimension reduction method via Rayleigh quotient optimization. *Ann. Statist.* **43** 1498-1534.

[21] Fan, J. and Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. *J. Roy. Statist. Soc. Ser. B* **70** 849–911.

[22] Fan, Y., Jin, J. and Yao, Z. (2013). Optimal classification in sparse Gaussian graphic model. *Ann. Statist.* **41** 2537–2571.

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

[24] Hall, P. and Jin, J. (2010). Innovated higher criticism for detecting sparse signals in correlated noise. *Ann. Statist.* **38** 1686-1732.

[25] Hsieh, C.-J., Dhillon, I. S., Ravikumar, P. K. and Sustik, M. A. (2011). Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems* 2330–2338.

[26] Huang, S., Jin, J. and Yao, Z. (2014). Supplemental material for ‘Partial Correlation Screening for estimating large precision matrix, with applications to classification’.

[27] Ingster, Y., Pouet, C. and Tsybakov, A. (2009). Classification of sparse high-dimensional vectors. *Phil. Trans. R. Soc. A* **367** 4427–4448.

[28] Jager, L. and Wellner, J. (2007). Goodness-of-fit tests via phi-divergences. *Ann. Statist.* **35** 2018-2053.

[29] Jin, J. and Ke, Z. (2015). Rare and weak effects in large-scale inference: methods and phase diagrams. *Statistica Sinica, To appear.*

[30] Jin, J., Zhang, C. H. and Zhang, Q. (2014). Optimality of Graphlet Screening in high dimensional variable selection. *J. Mach. Learn. Res.* **5** 2723–2772.

[31] Ke, Z., Jin, J. and Fan, J. (2014). Covariance Assisted Screening and Estimation. *Ann. Statist.* **42** 2202–2242.

[32] Li, R. and Zhu, L. (2012). Feature screening via distance correlation learning. *J. Amer. Statist. Assoc.* **107** 1129-1139.

[33] Mazumder, R. and Hastie, T. (2012). Exact covariance thresholding into connected components for large-scale graphical lasso. *J. Mach. Learn. Res.* **13** 723-736.

[34] Ravikumar, P., Wainwright, M. J., Raskutti, G. and Yu, B. (2011). High-dimensional covariance estimation by minimizing l_1-penalized log-determinant divergence. *Electron. J. Statist.* **5** 935–980.
[35] Seber, G. and Lee, A. (2003). *Linear Regression Analysis.* Wiley, New Jersey.
[36] Spiegelhalter, D. (2014). The future lies in uncertainty. *Science* **345** 264.
[37] Sun, T. and Zhang, C. H. (2012). Scaled sparse linear regression. *Biometrika* **99** 879–898.
[38] Vershynin, R. (2012). Introduction to the non-asymptotic analysis of random matrices. In *Compressed sensing* 210–268. Cambridge Univ. Press, Cambridge.
[39] Wasserman, L. and Roeder, K. (2009). High dimensional variable selection. *Ann. Statist.* **37** 2178–2201.
[40] Wittena, D. M., Friedmana, J. H. and Simon, N. (2011). New Insights and Faster Computations for the Graphical Lasso. *J. Comp. Graph. Stat.* **20** 892–900.
[41] Yousefi, M., Hua, J., Sima, C. and Dougherty, E. (2010). Reporting bias when using real data sets to analyze classification performance. *Bioinformatics* **26** 68-76.
[42] Yuan, M. (2010). High dimensional inverse covariance matrix estimation via linear programming. *J. Mach. Learn. Res.* **11** 2261–2286.
[43] Yuan, M. and Lin, Y. (2007). Model Selection and Estimation in the Gaussian Graphical Model. *Biometrika* **94** 19–35.
[44] Zhang, T. (2009). On the consistency of feature selection using greedy least squares regression. *J. Mach. Learn. Res.* **10** 555–568.
[45] Zhang, T. (2011). Adaptive Forward-Backward greedy algorithm for learning sparse representations. *IEEE Trans. Inf. Theory* **57** 4689–4708.
[46] Zhao, T., Liu, H., Roeder, K., Lafferty, J. and Wasserman, L. (2012). The huge package for high-dimensional undirected graph estimation in R. *The Journal of Machine Learning Research* **13** 1059–1062.
[47] Zhong, P., Chen, S.-X. and Xu, M. (2013). Test alternative to higher criticism for high dimensional means under sparsity and column wise dependence. *Ann. Statist.* **41** 2820-2851.

S. Huang and J. Jin
Department of Statistics
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
E-mail: shiqiong@stat.cmu.edu
E-mail: jiashun@stat.cmu.edu

Z. Yao
Department of Statistics and Applied Probability
National University of Singapore
Singapore 117546
E-mail: zhigang.yao@nus.edu.sg