Efficient Partitioning of Partial Correlation Networks

Keith Dillon[1][0000−0002−9495−6577]

University of New Haven, New Haven CT 06516, USA kdillon@newhaven.edu

Abstract. Partial correlation is a popular and principled metric for determining edges between nodes in a graph. However when the goal is to both estimate network connectivity from sample data and subsequently partition the result, methods such as spectral clustering can be applied much more efficiency and at larger scale. We derive a method that can similarly partition partial correlation networks directly from sample data. The method is closely related to spectral clustering, and can be implemented with comparable efficiency. Our results also provide new insight into the success of spectral clustering in many fields, as an approximation to clustering of partial correlation networks.

Keywords: Partial correlation · Spectral clustering · Graphical models · Graph partitioning

1 Introduction

Partial correlation is a natural choice for defining edges in a network. Unlike edges based on affinity or distance, the partial correlation removes the effect of indirect relationships [4] and measures the relationship based on the residual only (Fig. 1). Gaussian graphical models [9] directly relate partial correlation values to the inverse of the sample covariance matrix and to variable prediction via linear regression [6]. Such computations are not efficient for large networks, however; a separate regression problem must be solved for every variable. When the goal is clustering or partitioning of the network, the problem is usually approached, from a different perspective, with graphs formed simply via affinity or univariate correlation. This is used as a starting point for graph-theoretical approaches to partitioning such as spectral clustering [5], a continuous relaxation of the normalized cut algorithm for partitioning graphs.

In simple terms, spectral clustering is typically computed by applying k-means clustering to the rows of the last k eigenvectors of the Laplacian matrix $\mathbf{L}$. In cases where $\mathbf{L}$ must be computed first from a matrix of sample data $\mathbf{A}$, one can instead use the first k singular vectors of $\mathbf{A}$ [3]. As the first k singular vectors can be efficiently calculated for very large data sets, this provides a means to directly estimate the partitioning of a network from sample data. In such partitioning methods the relationships between nodes seems obfuscated under multiple approximations; first a crude univariate metric is used (Fig. 1); second the partitioning of this graph is approximated with a continuous relaxation.

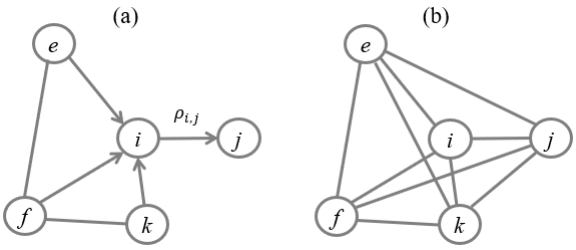

Fig. 1. Depiction of a network, where nodes e, f, and k are strongly interacting with i, but only i is interacting with j; (a) the partial correlation $\rho_{i,j}$ estimates edges by removing contributions from e, f, k, identifying the outlier status of j. (b) edges based on univariate tests such as affinity or distance result in a dense network where all nodes appear connected.

Spectral clustering has empirically shown promise, even in areas where partial correlation networks are preferred due to their sound statistical basis. In [3] it was found that a spectral approach was empirically more generalizable than simpler clustering methods, in terms of its ability to predict node relatedness for held-out data. In [1], it is shown that spectral clustering can approximate the community structure of a partial correlation network.

In this paper, we derive an algorithm for clustering of nodes in partial correlation networks directly from sampled data. The result can be viewed as a variant of spectral clustering with additional correction terms. We show that it can be computed with comparable efficiency to spectral clustering and demonstrate efficient computation for a dense network with roughly 100,000 nodes. We find that the success of spectral clustering noted above can be attributed to the small size of this correction term, particularly in situations where preprocessing steps such as filter are applied to the data.

2 Method

We model the data signal at the ith node as the zero-mean Gaussian random variable a_i. The partial correlation $\rho_{i,j}$ between a_i and a_j is the Pearson correlation between the residuals after regressing out all other nodes except i and j from both. These regression coefficients are defined as the solutions $\beta_{i,j}$ to the linear regression problem

$$a_i = \sum_{k \neq i} \beta_{i,k} a_k + \epsilon_i, \tag{1}$$

where a_i is the ith variable and ϵ_i is the residual. From these $\beta_{i,j}$ we can estimate $\rho_{i,j}$ as [6],

$$\rho_{i,j} = -\beta_{i,j} \sqrt{\frac{\sigma^{i,i}}{\sigma^{j,j}}}, \tag{2}$$

using the residual variances $\sigma^{i,i} = (Var(\epsilon_i))^{-1}$. A common alternative formulation exploits the symmetry of the partial correlation (i.e., that $\rho_{i,j} = \rho_{i,j}$ by definition) and uses the geometric mean to cancel the residual variances [7]

$$\rho_{i,j} = \text{sign}(\beta_{i,j})\sqrt{\beta_{i,j}\beta_{j,i}}. \tag{3}$$

This also has the advantage of enforcing symmetry in sample estimates. If the signs of $\beta_{i,j}$ and $\beta_{j,i}$ differ, $\rho_{i,j}$ is typically set to zero.

To provide sample formulations of the above estimates, we define $\mathbf{A}$ as a matrix containing data, with samples for a_k in the kth column $\mathbf{a}_k$ (which we will assume has been standardized). The vector form of the solution to Eq. (1) can be estimated as $\boldsymbol{\beta}_i = \mathbf{A}^{\dagger}_{-i}\mathbf{a}_i$, where $\mathbf{A}^{\dagger}_{-i}$ is the pseudoinverse of $\mathbf{A}$ after setting its ith column to zeros. Eq. (2) in terms of matrices becomes

$$\mathbf{P} = \mathbf{D_d}\mathbf{B}\mathbf{D_d}^{-1}, \tag{4}$$

where $\mathbf{D_d}$ is a diagonal matrix with $D_{i,i} = d_i = \|\mathbf{A}_{-i}\boldsymbol{\beta}_i - \mathbf{a}_i\|$, and $\mathbf{B}$ is a matrix with with $\boldsymbol{\beta}_i$ as columns (where the ith element of $\boldsymbol{\beta}_i$ is zero). $\mathbf{P}$ contains our sample-based estimates of the partial correlations, with $P_{i,j}$ describing the partial correlation between nodes i and j. Again, we can avoid calculating the residual variances using the method of Eq. (3) as follows,

$$\mathbf{P} = \text{sign}(\mathbf{B}) \odot (\mathbf{B} \odot \mathbf{B}^T)^{\circ\frac{1}{2}}, \tag{5}$$

using the Hadamard (element-wise) product $\odot$ and element-wise exponential $\circ\frac{1}{2}$, and where the sign function is taken element-wise.

2.1 Efficient Regularized Estimation of Partial Correlation

Thus far we have merely converted sample estimates of partial correlations into a matrix form. This direct formulation still requires a large amount of computation for each column of $\mathbf{B}$ (i.e., each node), by removing each column of $\mathbf{A}$ in turn, then taking the pseudoinverse of the result. In this subsection we will derive an efficient approach which only requires a single pseudoinverse for the entire network estimate. We will also provide a general form which incorporates regularization, often included as a heuristic data pre-processing step [2].

We incorporate regularization in the derivation by using the regularized pseudoinverse solution, $\boldsymbol{\beta}_i = (\mathbf{A}_{-i})^{\dagger}_{\lambda}\,\mathbf{a}_i$, where λ reflects the regularization parameter used in calculating the pseudoinverse (e.g., the singular value cutoff used). In order to remove the need for individual node pseudoinverses, we first define the matrix $\mathbf{B}^{(0)}$ with $\boldsymbol{\beta}^{(0)}_i$ as its ith column, $\mathbf{B}^{(0)} = \mathbf{A}^{\dagger}_{\lambda}\mathbf{A} = \mathbf{R}$, where $\mathbf{A}^{\dagger}_{\lambda}$ is the regularized pseudoinverse of $\mathbf{A}$, and where we have defined the (regularized) resolution matrix $\mathbf{R}$. For convenience we also define $\bar{\mathbf{A}}_{-i}$ as $\mathbf{A}_{-i}$ with the ith column (which is all zeros) removed. Then without loss of generality, we presume that $\mathbf{A} = (\bar{\mathbf{A}}_{-i}, \mathbf{a}_i)$, in order to simplify the equations (i.e., the order of the variables has been permuted so that the ith variable is last). With this we

can write $\mathbf{A}\boldsymbol{\beta}_i^{(0)} = (\bar{\mathbf{A}}_{-i}, \mathbf{a}_i)\,\boldsymbol{\beta}_i^{(0)} = \mathbf{a}_i$. We similarly define $\bar{\boldsymbol{\beta}}_i$ as $\boldsymbol{\beta}_i$ with the ith element removed (recall $\boldsymbol{\beta}_i$ was defined with a zero in the ith element). Then for an ℓ_2-regularized pseudoinverse, the solution for $\bar{\boldsymbol{\beta}}_i$ is

$$\bar{\boldsymbol{\beta}}_i = \bar{\mathbf{A}}_{-i}^T \left(\bar{\mathbf{A}}_{-i}\bar{\mathbf{A}}_{-i}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{a}_i \tag{6}$$

$$= \bar{\mathbf{A}}_{-i}^T \left(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I} - \mathbf{a}_i\mathbf{a}_i^T\right)^{-1}\mathbf{a}_i \tag{7}$$

We employ the matrix inversion lemma to get

$$\bar{\boldsymbol{\beta}}_i = \bar{\mathbf{A}}_{-i}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{a}_i - \frac{\bar{\mathbf{A}}_{-i}^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{a}_i\mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{a}_i}{-1 + \mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I})^{-1}\mathbf{a}_i} \tag{8}$$

Meanwhile, the least-squares solution for $\boldsymbol{\beta}_i^{(0)}$ is,

$$\boldsymbol{\beta}_i^{(0)} = \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{a}_i = \begin{pmatrix}\bar{\mathbf{r}}_i^{(\lambda)} \\ R_{i,i}^{(\lambda)}\end{pmatrix}, \tag{9}$$

where $\bar{\mathbf{r}}_i$ is the ith column of the regularized resolution matrix with the ith element removed, and $R_{i,i}$ is the (i,i)th element. Utilizing these definitions in Eq. (8) gives,

$$\bar{\boldsymbol{\beta}}_i = \left(\frac{1}{1 - R_{i,i}}\right)\bar{\mathbf{r}}_i \tag{10}$$

To write the matrix version of this relation between $\mathbf{B}$ and $\mathbf{R}$, we form the matrix $\mathbf{R}_{-d}$ defined as $\mathbf{R}$ with the values on the diagonal set to zero, and perform the scaling as $\mathbf{B} = \mathbf{R}_{-d}\mathbf{D_s}$ where $\mathbf{D_s}$ is a diagonal matrix with $D_{i,i} = s_i = (1 - R_{i,i})^{-1}$. Inputting this into Eq. (4) gives

$$\mathbf{P}^{(a)} = \mathbf{D_d}\mathbf{R}_{-d}\mathbf{D}_s\mathbf{D_d}^{-1}. \tag{11}$$

We can also write the $\mathbf{d}$ vector (the diagonal of $\mathbf{D_d}$) as

$$d_i = \|\mathbf{A}_{-i}\boldsymbol{\beta}_i - \mathbf{a}_i\| = \left|\tfrac{1}{1-R_{i,i}}\right| \left\|\mathbf{A}\left(\mathbf{A}^\dagger\mathbf{a}_i - \mathbf{e}_i\right)\right\|, \tag{12}$$

where $\mathbf{e}_i$ is the i column of the identity matrix.

We refer to Eq. (11) as the asymmetric version, as we ignore possible asymmetries $P_{ij} \neq P_{ji}$. Alternatively, we can extend the symmetric version of Eq. (5) by plugging in $\mathbf{B} = \mathbf{R}_{-d}\mathbf{D_s}$, to get,

$$\mathbf{P}^{(s)} = \text{sign}(\mathbf{1}\mathbf{s}^T) \odot (\mathbf{s}\mathbf{s}^T)^{\circ\frac{1}{2}} \odot \mathbf{R}_{-d}. \tag{13}$$

where $\mathbf{s}$ is the vector with elements $s_i = (1 - R_{i,i})^{-1}$. In this version we set $P_{i,j}$ equal to zero when $\text{sign}(s_i) \neq \text{sign}(s_j)$.

1. Choose number of clusters K and initialize cluster centers $\mathbf{c}_k,\ k = 1, ..., K$;
while *Convergence criterion not met* **do**
  2. Label each column as belonging to nearest cluster center: $l_k = \arg\min_i D_{ik}$, using minimum over distance D_{ik} between every column $\mathbf{p}_k$ and every cluster center $\mathbf{c}_i$;
  3. Recalculate cluster centers as mean over data columns with same label: $\mathbf{c}_i = \frac{1}{|S_i|}\sum_{j\in S_i}\mathbf{a}_j$, where $S_i = \{k|l_k = i\}$;
end

Algorithm 1: k-means clustering of columns of $\mathbf{P}$.

2.2 Efficient Clustering of P

Next we will show how to partition the partial correlation network directly using the raw data $\mathbf{A}$. A basic k-means clustering algorithm which could be used for clustering the columns of $\mathbf{P}$ is given in Algorithm 1. Of course, for a large network, $\mathbf{P}$ will be extremely large as it has one entry per edge, so N nodes results in a matrix of size $N \times N$. However, as we have eliminated the need for node-specific pseudoinverses for each column, we can compute columns on-the-fly as needed inside the clustering loop, as in

$$\begin{aligned}\mathbf{p}_i^{(a)} &= \frac{1}{d_i(1-R_{i,i})}\mathbf{d}\odot\mathbf{r}_{-i}\\ &= \mathbf{d}\odot\left(\mathbf{A}^\dagger\alpha_i\mathbf{a}_i - R_{i,i}\alpha_i\mathbf{e}_i\right),\end{aligned} \tag{14}$$

where we have defined $\alpha_i = [d_i(1-R_{i,i})]^{-1}$ and $\mathbf{r}_{-i}$ is the ith column of $\mathbf{R}_{-d}$. We can precompute one pseudoinverse, $\mathbf{A}^\dagger$, the $\mathbf{d}$ vector from Eq. (12), and the diagonal terms $R_{i,i}$ of the resolution matrix. For the symmetric version we compute columns on the fly with a similar form,

$$\begin{aligned}\mathbf{p}_i^{(s)} &= \mathrm{sign}(s_i)(s_i\mathbf{s})^{\circ\frac{1}{2}}\odot\mathbf{r}_{-i}\\ &= |\mathbf{s}|^{\circ\frac{1}{2}}\odot(\mathbf{A}^\dagger\sigma_i\mathbf{a}_i - R_{i,i}\sigma_i\mathbf{e}_i),\end{aligned} \tag{15}$$

where $\sigma_i = \mathrm{sign}(s_i)|s_i|^{\frac{1}{2}} = \mathrm{sign}(1-R_{i,i})|1-R_{i,i}|^{-\frac{1}{2}}$, which can also be precomputed. Generally we write either Eq. (15) or Eq. (14) as

$$\mathbf{p}_i = \mathbf{z}\odot(\mathbf{A}^\dagger\zeta_i\mathbf{a}_i - R_{i,i}\zeta_i\mathbf{e}_i), \tag{16}$$

for appropriate definitions of $\mathbf{z}$ and $\boldsymbol{\zeta}$.

The squared distances between a given center $\mathbf{c}_i$ and a column $\mathbf{p}_k$ of $\mathbf{P}$ can be calculated as

$$\begin{aligned}D_{ik}^2 &= \|\mathbf{c}_i - \mathbf{p}_k\|_2^2\\ &= \mathbf{c}_i^T\mathbf{c}_i + \mathbf{p}_k^T\mathbf{p}_k - 2\mathbf{c}_i^T\mathbf{p}_k.\end{aligned} \tag{17}$$

Since we are only concerned with the class index i of the cluster with the minimum distance to each column, we can compute the labels as

$$\begin{aligned} l_k &= \arg\min_i D_{ik}^2 \\ &= \arg\min_i \left\{ \mathbf{c}_i^T \mathbf{c}_i - 2(\mathbf{c}_i \odot \mathbf{z})^T (\mathbf{A}^\dagger \zeta_k \mathbf{a}_k - R_{k,k} \zeta_k \mathbf{e}_k) \right\}. \end{aligned} \tag{18}$$

By forming a matrix $\mathbf{C_z}$ with weighted cluster centers $\mathbf{c}_i \odot \mathbf{z}$ as columns, and a weighted data matrix $\mathbf{A}_\zeta$ with $\zeta_i \mathbf{a}_i$ as columns, we can efficiently compute the first part of the cross term for all i and k as $(\mathbf{C_z}^T \mathbf{A}^\dagger)\mathbf{A}_\zeta$, a K by n matrix. The second part of the cross term can be computed by (element-wise) multiplying each row of $\mathbf{C_z}^T$ by a vector who's kth element is $R_{k,k}\zeta_k$. Similar tactics can be used to efficiently compute the mean over columns in each cluster, as

$$\begin{aligned} \mathbf{c}_i &= \frac{1}{|S|} \sum_{j \in S} \mathbf{p}_j \\ &= \frac{1}{|S|} \mathbf{z} \odot \left(\mathbf{A}^\dagger \sum_{j \in S} \zeta_j \mathbf{a}_j - \sum_{j \in S} R_{j,j} \zeta_j \mathbf{e}_j \right) \end{aligned} \tag{19}$$

So in general, we see that clustering of $\mathbf{P}$ can be implemented whenever we are able to to implement k-means clustering of the original dataset $\mathbf{A}$, taking roughly double the storage space and computational resources.

In [2] we derived a similar algorithm for clustering columns of the matrix $\mathbf{R}$; in that case $l_k = \arg\min_i \left\{ \mathbf{c}_i^T \mathbf{c}_i - 2\mathbf{c}_i^T \mathbf{A}^\dagger \mathbf{a}_k \right\}$, and $\mathbf{c}_i = \frac{1}{|S|} \mathbf{A}^\dagger \sum_{j \in S} \mathbf{a}_j$. In [3] we showed that this algorithm could be viewed as a variant of spectral clustering. So the corrections relating partial correlation clustering and spectral clustering are due to the ζ_j and $\mathbf{z}$ scaling factors, plus the $\mathbf{e}_j$ correction terms.

3 Results

First we produced an artificial dataset with a two-dimensional correlation structure. We generated 3000 samples for each of 300 random variables, with the kth random variable defined as,

$$a_k = x_k s_0 + (1 - x_k) s_1 + y_k t_0 + (1 - y_k) t_1 + \sigma n_k,$$

where s_0, s_1, t_0, and t_1 and n_k are independent unit-variance random variables, and $x_k \in [0, 1]$ and $y_k \in [0, 1]$ are randomly-chosen points. Effectively, for variables with (x_k, y_k) nearer to each other, the signals a_k are more correlated. We expect a simple partitioning of the x, y domain as a result of clustering this data. Simulated results are given in Fig. 2, using four clusters. By choosing the regularization parameter at a level equivalent to choosing the first two nontrivial eigenvectors, we achieved similar results for spectral and partial correlation clustering even in the presence of high noise, while basic k-means clustering of the noisy signals failed.

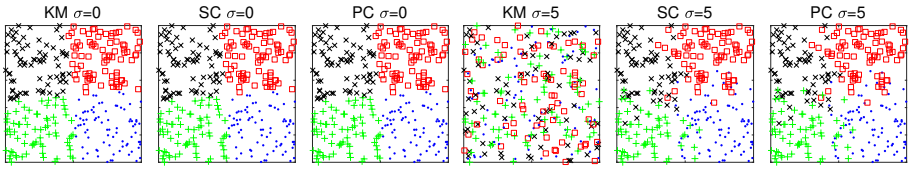

Fig. 2. Simulated results for k-means(KM), spectral clustering (SC), and partial correlation clustering (PC) of data with two-dimensional correlation structure, for noise levels $\sigma = 0$ and 5.

Next we considered a high-dimensional real dataset. Fig. 3 demonstrates the algorithm applied to functional Magnetic Resonance Imaging (fMRI) scans for a subject from the Human Connectome Project [8], compared to other clustering approaches. The data was preprocessed by applying spatial smoothing with a 5mm kernel, and regularization was used to achieve a cutoff of 30 percent of singular values. The data contains 96854 time series with 1200 time samples each, resulting in a data matrix $\mathbf{A}$ of size 1200×96854. Each column contains a time-series describing the blood-oxygen-level dependent (BOLD) activity in one voxel of the brain, so a network formed by comparing these signals provides an estimate of the functional connectivity of the brain. A clustering of this network would produce an estimate of the modularity of function in the brain. The network describing the relationships between all pairs of voxels, however, would require a $\mathbf{P}$ matrix of size 96854×96854, which is far too large to fit in RAM. However the limited rank of this matrix means we only need store the 96854×1200 pseudoinverse. In this case the clustering algorithm took 9 seconds on a desktop computer. We see that clustering of $\mathbf{P}$ produces much more

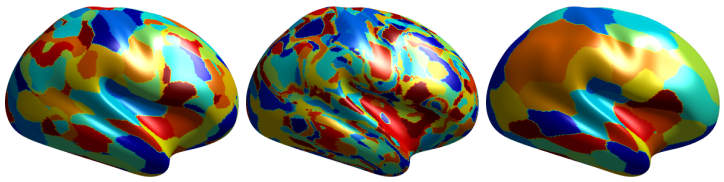

Fig. 3. Clustering functional MRI data for single subject from the HCP project into 100 clusters; k-means clustering of the original time series (left); clustering of univariate correlations between time series (middle); clustering of partial correlations between time-series (right).

modular segmentation of the regions of the brain, particularly compared to the conceptually-similar approach of clustering the network of univariate correlations of the data instead.

We tested the difference between spectral clustering and partial correlation for this dataset. Using identical random initial clusters for both methods, we

plot the percentage of nodes which differ in the final clustering results in Fig. 4, as increasing amounts of spatial smoothing are applied. With spatial smoothing,

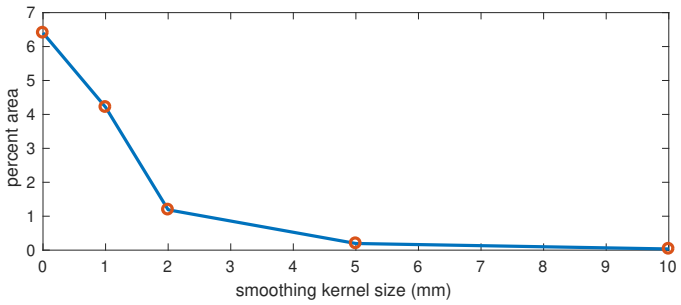

Fig. 4. Plot of difference between clustering partial correlation network versus spectral clustering, showing close agreement between the methods, particularly with spatial smoothing. Spectral clustering in this case uses the covariance matrix of the data as weighted adjacency matrix.

a common preprocessg step in many applications, the removal of the diagonal terms will have less effect, as for example, the same information is increasingly included in the neighboring variables.

4 Discussion

We derived an efficient approach to partitioning partial correlation networks and demonstrated the approach on neuroimaging data, where we found a close similarity to spectral clustering. This suggests another perspective on the success of spectral clustering methods, as an approximation to clustering of a Gaussian graphical model for the data. A benefit of the proposed approach is its principled basis as a direct estimate for a partial correlation network. As our simulation shows, the benefit of spectral clustering over basic k-means can be viewed, at least in a low-dimensional setting, as resulting from the regularization effect of truncating the eigenvectors. With the experimental data, we also see a close agreement with spectral clustering in higher dimensions, particularly when spatial smoothing is employed. In terms of numerical efficiency, our approach provides a more efficient calculation compared to the brute-force approach of solving a separate regression problem for each node.

The drawbacks of the proposed approach include the moderately increased computational effort, and the potential need to address asymmetric signs in the partial correlation matrix. Though our approach to address the latter (setting them to zero based on a sign test) is the approach commonly used in bioinformatics. There are a number of potential extensions to the approach. Instead of a simple k-means stage, we might apply a more sophisticated clustering algorithm

such as fuzzy or hierarchical clustering. Also we could extend the distance calculation to other and more sophisticated statistics or more sophisticated statistical tests.

5 Appendix: matlab implementation of clustering

In this appendix we provide efficient Matlab code for performing partial correlation clustering.

```
A = randn(500,100000); % simulate data matrix
lambda = 1; % regularization parameter
k = 100; % number of clusters
[rows_A,cols_A] = size(A);

% standardize data columns
A = bsxfun(@minus,A,mean(A));
A = bsxfun(@times,A,1./sum(A.^2).^.5);

% compute diagonal of R via sum of squared eigenvectors
[uA,sA,vA] = svd(A,'econ');
r = sum(vA(:,1:rank(A)).^2,2)';
r = r(:);

% compute pseudoinverse efficiently (assuming fewer rows than columns)
iA_lambda = A'*inv(A*A'-lambda*eye(rows_A));

% compute scaling vectors (symmetric version)
s = 1./(1-r(:));
z = abs(s(:)).^.5;
zeta = sign(s).*abs(s(:)).^.5;
Az = bsxfun(@times,A,z(:)'); % precompute scaled version

% randomly assign columns to clusters initially
c = ceil(rand(cols_A,1)*k);
n_change = inf
while (n_change>0) % clustering loop
    M = sparse(1:cols_A,c,1,cols_A,k,cols_A); % cols of M are masks of clusters
    M = bsxfun(@times, M, 1./sum(M,1));  % now M is averaging operator
    P_c_1 = iA_lambda*(Az*M);  % first part of cluster center calc
    P_c_2 = bsxfun(@times,M,r.*zeta); % second park (peak removal)
    P_c = bsxfun(@times,P_c_1-P_c_2,z(:)); % cluster centers
    Pz2_c = sum(P_c.^2,1); % squared term from distance
    Cz = bsxfun(@times,P_c,z(:)); % weighted cluster centers
    D_ct1 = (Cz'*iA_lambda)*Az; % first part of cross-term
    D_ct2 = bsxfun(@times,Cz',r'.*zeta(:)'); % second part of cross term
    D_ct = D_ct1-D_ct2; % cross-term
    Dz = bsxfun(@minus,D_ct,.5*Pz2_c'); % dist metric (sans unnecessary term)
    c_old = c;
    [D_max,c(:)] = max(Dz,[],1); % c is arg of max
```

```
    n_change = sum(c~=c_old);
    disp(n_change);
end;
```

References

1. Brownlees, C., Gudmundsson, G.S., Lugosi, G.: Community Detection in Partial Correlation Network Models. Journal of Business & Economic Statistics **0**(0), 1–11 (Jul 2020).
2. Dillon, K., Wang, Y.P.: A regularized clustering approach to brain parcellation from functional MRI data. In: Wavelets and Sparsity XVII. vol. 10394, p. 103940E. International Society for Optics and Photonics (Aug 2017).
3. Dillon, K., Wang, Y.P.: Resolution-based spectral clustering for brain parcellation using functional MRI. Journal of Neuroscience Methods **335**, 108628 (Apr 2020).
4. Ellett, F.S., Ericson, D.P.: Correlation, partial correlation, and causation. Synthese **67**(2), 157–173 (May 1986).
5. von Luxburg, U.: A tutorial on spectral clustering. Statistics and Computing **17**(4), 395–416 (Dec 2007).
6. Pourahmadi, M.: Covariance Estimation: The GLM and Regularization Perspectives. Statistical Science **26**(3), 369–387 (Aug 2011).
7. Schäfer, J., Strimmer, K.: A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. Statistical Applications in Genetics and Molecular Biology **4**(1) (2005).
8. Van Essen, D.C., Ugurbil, K., Auerbach, E., Barch, D., Behrens, T.E.J., Bucholz, R., Chang, A., Chen, L., Corbetta, M., Curtiss, S.W., Della Penna, S., Feinberg, D., Glasser, M.F., Harel, N., Heath, A.C., Larson-Prior, L., Marcus, D., Michalareas, G., Moeller, S., Oostenveld, R., Petersen, S.E., Prior, F., Schlaggar, B.L., Smith, S.M., Snyder, A.Z., Xu, J., Yacoub, E., WU-Minn HCP Consortium: The Human Connectome Project: a data acquisition perspective. NeuroImage **62**(4), 2222–2231 (Oct 2012).
9. Whittaker, J.: Graphical Models in Applied Multivariate Statistics. Wiley Publishing (2009)