ECE 602 Project: Implementation of "Network Inference via the Time-Varying Graphical Lasso"


```
Authors: David Hallac, Youngsuk Park, Stephen Boyd, Jure Leskovec
```

- Students: Nima Abbasi Firoozjah, Abul Hasan Fahad, and Mahta Ramezanian
- Institution: University of Waterloo, Ontario, Canada

- Session: Winter 2019
- Instructor: Prof. Oleg Michailovich

Contents

1. Background

Many practical systems can be modeled as a network of interacting entities or nodes. Observational data can be used to infer relationships among these nodes which are subject to temporal changes. An approach to study such networks is the usage of an underlying time-varying inverse covariance matrix.

In this paper, the time-varying network inference approach is formulated as a convex optimization problem. Two primary contributions are proposed: 1) Formally defining this problem, as time-varying graphical lasso (TVGL), and 2) Deriving a scalable optimization method to solve it.

2. Problem Definition

Given a sequence of multivariate observations, the TVGL estimates the inverse covariance $\Sigma^{-1}(t)$ of the data to simultaneously achieve three goals: (1) matching the empirical observations, (2) sparsity, and (3) temporal consistency.

3. Convex Optimization Formulation

A convex optimization problem can be formulated for $\Theta_i = \Sigma(t_i)^{-1}$, one for each t_i . We can determine Θ = $(\Theta_1, \Theta_1, ..., \Theta_T)$ (our estimated inverse-covariance matrices at times $t_1, t_2, ..., t_T$)

by solving the following minimization problem

$$minimize_{\Theta \in S^P_{++}} \quad \sum_{i=1}^{T} -l_i(\Theta_i) + \lambda||\Theta_i||_{od,1} + \beta \sum_{i=2}^{T} (\Theta_i - \Theta_{i-1})$$

where

$$l_i(\Theta_i) = n_i(logdet(\Theta_i) - \mathbf{Tr}(S_i\Theta_i)$$

with $\Theta \in S^P_{++}$ (positive-definitie), $||\Theta||_{od,1}$ is the seminorm of Θ, S in the empirical-covariance $\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$, n is the number of observations and x_i are different samples.

$\lambda \geq 0$

$\beta \geq 0$

$\psi(\Theta_i - \Theta_{i-1})$ is a convex penalty function

4. Choice of Penalty Functions

In order to accommodate different network behaviours, five different penalty functions ψ are proposed . They are:

i. A few edges changing at a time: Useful when only a few edges are expected to change at a given time.

ii. Global restructuring: this function seeks the exact time of the major change in the underlying covariance matrix (useful for event detection and time-series segmentation)

iii. Smoothly varying over time: This penalty function is useful for the objectives with a smoothly varying estimate over time.

iv. Block-wise restructuring: Useful when expecting a subset of nodes to changes their internal edge structure suddenly, while the rest of the network remains unchanged.

v. Perturbed node: The perturbed node penalty function is defined as $\psi(X) = min \sum ||[V]_j||_2$ such that $V : V + V^T = X$. This penalty function is useful when single nodes are re-locating themselves to a new set of neighbors within the network.

In this work, the perturbed node penalty function will be investigated.

5. Solution: Application of ADMM

In the paper, Alternating direction method of multipliers (ADMM) based solution was proposed. In ADMM, the problem falls into a series of subproblems, using a message-passing algorithm to converge on the globally optimal solution.

To split the Problem (section-3) into a separable form, define a consensus variable Z = $[Z_0, Z_1, Z_2]$ = $[(Z_{1,0}, ..., Z_{T,0}), (Z_{1,1}, ..., Z_{T-1,1}), (Z_{2,2}, ..., Z_{T,2})]$

With this, Problem (section-3) can be re-written as its equivalent problem,

$$minimize \quad \sum_{i=1}^{T} -l_i(\Theta_i) + \lambda||\Theta_i||_{od,1} + \beta\sum_{i=2}^{T} \psi(Z_{i,2} - Z_{i-1,1})$$

$$subject \quad to \quad Z_{i,0} = \Theta_i, \Theta_i \in S_{++}^{P} \quad for \quad i = 1, ..., T \quad and$$

$$(Z_{i-1,1}, Z_{i,2}) = (\Theta_{i-1}, \Theta_i) \quad for \quad i = 2, ..., T$$

The corresponding augmented Lagrangian can be written as-

where U = $[U_0, U_1, U_2]$ = $[(U_{1,0}, ..., U_{T,0}), (U_{1,1}, ..., U_{T-1,1}), (U_{2,2}, ..., U_{T,2})]$ is the scaled dual variable and

ρ is the ADMM penalty parameter

Steps of Iterative Updates (k denotes the iteration number):

i) $Theta$ Update:

The Θ is updated seprately for each Θ_i, which can then be solved in parallel:

where (a) holds for

and b holds due to the symmetry of Θ_i

This can be rewritten as a proximal operator

Since $\frac{A+A^T}{2}$ is symmetric, this has an analytic solution where QDQ^T is the eigen-decomposition of $\eta^{-1}\frac{(A+A^T)}{2}-S_i$

ii) Z update:

Z_0 -Update: Each $Z_{i,0}$ can be written as the proximal operator of the $l_{od,1}$ -norm, which has a known closed-form solution

Z_1, Z_2 -Update:

$Z_{i-1,1}$ and $Z_{i,2}$ are coupled together in the augmented Lagrangian, so they must be jointly updated. In order to derive the closed-form solution, it is defined as

Now solving for a separate update for each (Z_1, Z_2) pair,

For analytical solution, the following formulation can be derived-

where

Defining

the following simplification can be obtained

where ϕ is the column-norm for l_1, l_2, l_2^2, and $l_{\inf}$.

(Z_1, Z_2) -Update for Perturbed Node Penalty:

This new problem is solved by deriving a second ADMM algorithm. Here, in each iteration of our original ADMM solution, we now call this second ADMM solver. In order to avoid notational conflict, we denote the minimization variables $(Z_{i-1,1}, Z_{i,2})$ as (Y_1, Y_2).

introducing an additional variable $V = W^T$, the augmented Lagrangian $\mathcal{L}_\rho$ becomes

where $(\tilde{U}_1, \tilde{U}_2)$ is the scaled dual variable and ρ is the same ADMM penalty parameter as outer ADMM. At the l-th iteration, the three steps in the ADMM update are as follows:

which has the following closed form solution for the j th column.

With

$A = \frac{Y_1^1+Y_2^1-W^l+\tilde{U_1^l}+((W^l)^T-\tilde{U_2^l})^T}{2}$

where $C = [I \quad -I \quad I]$ and $D = V^l + \tilde{U_1^l}$

6. Proofs and Critical Review

In the following, we provide our proofs of non-trivial equations in the paper and also the corrections to the formulas. We examined the published version of the paper and it's identical to the version we received for our project so we contacted the authors regarding our concerns. Most of our suspicions were confirmed by the authors of the paper and they are mentioned below.

For the Θ-\textit{Update} step, the A matrix is defined as following

$$A = \frac{Z_{i,0}^k + Z_{i,1}^k + Z_{i,2}^k - U_{i,0}^k - U_{i,1}^k - U_{i,2}^k}{3}$$

Also, η is defined as $\eta = \frac{n_i}{3\rho}$. However, we can observe that in the augmented Lagrangian expression there are only two terms including θ_1 and θ_T, therefore we should modify the A matrix and η for $i = 1$ as follows

$$A = \frac{Z_{1,0}^k + Z_{1,1}^k - U_{1,0}^k - U_{1,1}^k}{2}, \quad \eta = \frac{n_i}{2\rho}$$

The correction for $i = T$ is as follows

$$A = \frac{Z_{T,0}^k + Z_{T,2}^k - U_{T,0}^k - U_{T,2}^k}{2}, \quad \eta = \frac{n_i}{2\rho}$$

In order to prove the aforementioned claim, we just need to prove that

$$\alpha = ||M - N||_F^2 + ||M - P||_F^2 - 2||M - A||_F^2$$

is constant in terms of M matrix where $A = \frac{N+P}{2}$. Using trace-formulation of the Frobenius-norm, we have

$$\alpha = \mathbf{Tr}((M-N)^T(M-N) + (M-P)^T(M-P) - 2(M-A)^T(M-A))$$

which can be simplified to

$$\alpha = \mathbf{Tr}(N^TN + P^TP - \frac{1}{2}(N+P)^T(N+P))$$

and we can simply observe that this term is independent from M. (end of the proof)

There are a few steps that need to be clarified since there are no proofs provided for them in the paper. For the $Perturbed\ Node\ Penalty$, the Z_1 and Z_2 are computed at each iteration by minimizing a $proximal$ function defined using $Perturbed\ Node\ Penalty$ function which include a constrained minimization itself. Merging these two minimization problems into one, allows us to write the augmented Lagrangian in the inner ADMM as it is in the paper with the extra equality constraint which is $V = W^T$.

For $(Z_1, Z_2) - Update$ using $Perturbed\ Mode\ Penalty$, again there is an A matrix defined as follows

$$A = \frac{Y_1^1 - Y_2^1 - W^l - \tilde{U}_1^l + ((W^l)^T - \tilde{U}_2^l)^T}{2}$$

However, it can be easily observed that W^l will be canceled out in the expression. We suspected that this might be a mistake and in order to make sure, we contacted the authors and realized that the last $transpose$ operation is an error in the formula and A should be defined as

$$A = \frac{Y_1^1 - Y_2^1 - W^l - \tilde{U}_1^l + ((W^l)^T - \tilde{U}_2^l)}{2}$$

Moreover, in the published paper, for the inner ADMM, the equation defining the $(l+1)$-th iteration of the j-th column of V is as follows

which seems to be incorrect, as well. We can confirm this suspicion by noticing that V is defined by a $proximal$ equation which is similar to the one in **Group Lasso L2 Penalty** section. As a result, we can see that a β is missing and the formula should be modified as follows

Also, in the (b)-step of the inner ADMM for $(Z_1, Z_2) - Update$, the V matrix from the previous iteration is used which seems to be a mistake, too. Our MATLAB code converged after fixing this issue.

7. Dataset Description

In this project, closing stock prices for Apple, Google, Amazon, Intel, Boeing, and FedEx is being observed on a daily basis over a few months. A network of relationships is then built based on this historic dataset which changes over time. One would expect the correlations between these companies to change smoothly over time unless an unusual event interrupts the trend. By perturbation node penalty approach, our goal is to identify these events, their role in the network dynamics, and finally the state of relations i.e. the structure of the network. These findings are useful in predicting prices and financial trends. A harsh change in the dependencies may correspond to a sudden shift in the correlations, indicating the perturbed node. Ergo, the perturbed node penalty is useful in identifying the nodes with a large single-node effect on the network.

8. Results

Our results turned out very similar to the ones in the paper as it was expected. in the plot below, one can see the temporal deviation of the stock network. The large spike in January indicates a sudden, non-smooth change in the correlation network of the companies, in this case, one can deduce that some unusual event happened on that time which changed the network structure drastically. Here, this abnormality is due to the introduction of the first iPad version by Apple Inc. This announcement yields a large change in the relationship of Apple Inc. with other software and shipping companies.

9. Conclusion and Discussion

TVGL proved to be an efficient method in recognizing temporal dependencies in dynamic networks. The authors suggest that this method applies in various scales which is open to further investigations. This model can be more generalized for the case with higher degrees of variation over time, e.g. mean and/or covariance. Also, this method can be used to more general cases where the structural changes are not immediate.

10. The Code

```
clc
close all
clear

% delete(gcp);
% parpool(72);

tic;
```

Data loading and parameter definitions

```
% loading data

temp = load('AAPL_AMZN_BA_FDX_INTC_MSFT.mat');

stock_data = temp.stock_data;

n_days = 244;

% parameters

rho = 2;

lambda = 0.5;

beta = 50;

eps = 3e-3;

epsAbs = 7e-4;
epsRel = 7e-4;
```

```
samplePerStep = 3;

center = 102;

T = length(center - 3*samplePerStep : samplePerStep : center + 13*samplePerStep) - 1;
```

```
Error using load
'AAPL_AMZN_BA_FDX_INTC_MSFT.mat' is not found in the current folder or on the MATLAB path, but exists in:
    C:\Users\Mahta\Documents\Courses\Optimization

Change the MATLAB current folder or add its folder to the MATLAB path.

Error in final (line 309)
temp = load('AAPL_AMZN_BA_FDX_INTC_MSFT.mat');
```

empirical covariances

```
S_matrices = rand(6, 6, T);

for i = 1:T

    i_beg = (i-1)*samplePerStep + center - 3*samplePerStep;
    i_end = i_beg + samplePerStep - 1;

    current_data = stock_data(i_beg:i_end, :);
    mu = mean(current_data);                                                                 % centering the data in each timestamp

    S_matrices(:, :, i) = 1/(samplePerStep) * (current_data - mu)' * (current_data - mu);  % computing the new epirical covariance matrix


end
```

Main ADMM

```
% initialization

Z0 = rand(6, 6, T);
Z1 = rand(6, 6, T);
Z1(:, :, T) = zeros(6, 6);
Z2 = rand(6, 6, T);
Z2(:, :, 1) = zeros(6, 6);
```

```
Theta = rand(6, 6, T);

U0 = rand(6, 6, T);
U1 = rand(6, 6, T);
U1(:, :, T) = zeros(6, 6);
U2 = rand(6, 6, T);
U2(:, :, 1) = zeros(6, 6);

r = 10;
s = 10;

epsPri = 1;
epsDual = 1;

cntr = 0;

while r > epsPri || s > epsDual

    cntr = cntr + 1;
    fprintf('counter = %d \n\n\n', cntr);


    Z0_old = Z0;
    Z1_old = Z1;
    Z2_old = Z2;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% (a) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    for i = 1:T

       Z_0 = Z0(:, :, i);
       Z_1 = Z1(:, :, i);
       Z_2 = Z2(:, :, i);
       U_0 = U0(:, :, i);
       U_1 = U1(:, :, i);
       U_2 = U2(:, :, i);

       if i == 1 || i==T

           A = (Z_0 + Z_1 + Z_2 - U_0 - U_1 - U_2)/2;
           eta = samplePerStep/(2*rho);

       else

           A = (Z_0 + Z_1 + Z_2 - U_0 - U_1 - U_2)/3;
           eta = samplePerStep/(3*rho);
```

```
        end

        temp_mat = 1/eta * (A+A')/2 - S_matrices(:, :, i);

        [Q, D] = eig(temp_mat);

        theta = eta/2 * Q * (D + sqrtm(D*D + 4/eta*eye(6))) * Q';

        Theta(:, :, i) = theta;                                          % updating Theta(i)

        AA = theta + U_0;

        Z0(:, :, i) = (abs(AA) > lambda/rho) .* (abs(AA) - lambda/rho) .* sign(AA);      % updating Z0(i)

    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%% (b) %%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    eta_in = beta/(2*rho);

    for i = 2 : T

        theta = Theta(:, :, i);
        theta_p = Theta(:, :, i-1);

        u1 = U1(:, :, i-1);
        u2 = U2(:, :, i);

        V = rand(6,6);
        U1_tilde = rand(6,6);
        U2_tilde = rand(6,6);
        Y1 = rand(6,6);
        Y2 = rand(6,6);
        W = rand(6,6);

        cntr_in = 0;

        r_in = 10;

        eps_in = 0.001;
```

inner ADMM for perturbed node penalty updating (Z1, Z2)

```
    while r_in > eps_in

        cntr_in = cntr_in + 1;
        fprintf('inner counter = %d \n', cntr_in);

        V_old = V;
        W_old = W;
        Y1_old = Y1;
        Y2_old = Y2;
        U1_tilde_old = U1_tilde;
        U2_tilde_old = U2_tilde;

        %%%%%%%%%%% (a) %%%%%%%%%%%%

        A = (Y1 - Y2 - W - U1_tilde + (W' - U2_tilde))/2;

        for j = 1:6

            if  norm(A(:, j)) <= eta_in

                V(:, j) = 0;

            else

                V(:, j) = ( 1 - eta_in/norm(A(:, j)) ) * A(:, j);

            end
        end

        %%%%%%%%%%% (b) %%%%%%%%%%%%

        C = [eye(6), -eye(6), eye(6)];
        D = V + U1_tilde;

        temp_mat = (C'*C + eye(18)) \ ([(V+U2_tilde)'; (theta_p+u1); (theta+u2)] - C'*D);

        W = temp_mat(1:6, :);
        Y1 = temp_mat(7:12, :);
        Y2 = temp_mat(13:18, :);

        %%%%%%%%%%% (c) %%%%%%%%%%%%

        U1_tilde = U1_tilde + (V+W)-(Y1-Y2);
        U2_tilde = U2_tilde + (V-W');

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(computing stopping criteria) for inner ADMM

```
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        r_in = norm(V_old - V, 'fro') + norm(Y1_old - Y1, 'fro') + norm(Y2_old - Y2, 'fro') + norm(W_old - W, 'fro') + ...
               norm(U1_tilde_old - U1_tilde, 'fro') + norm(U2_tilde_old - U2_tilde, 'fro');


    end

    Z1(:, :, i-1) = Y1;                                                     % updating Z1(i-1)
    Z2(:, :, i) = Y2;                                                       % updating Z2(i)

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% (c) %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


U0 = U0 + Theta - Z0;                                                       % updating U0
U1(:, :, 1:T-1) = U1(:, :, 1:T-1) + Theta(:, :, 1:T-1) - Z1(:, :, 1:T-1);     % updating U1
U2(:, :, 2:T) = U2(:, :, 2:T) + Theta(:, :, 2:T) - Z2(:, :, 2:T);             % updating U2



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%% (computing stopping criteria) for the main ADMM %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

r_temp_1 = Theta - Z0;
r_temp_2 = Theta(:, :, 1:T-1) - Z1(:, :, 1:T-1);
r_temp_3 = Theta(:, :, 2:T) - Z2(:, :, 2:T);

s_temp_1 = Z0 - Z0_old;
s_temp_2 = Z1(:, :, 1:T-1) - Z1_old(:, :, 1:T-1);
s_temp_3 = Z2(:, :, 2:T) - Z2_old(:, :, 2:T);

epsPri = sqrt(6) * epsAbs + 0.0001;
epsDual = sqrt(T) * epsAbs + 0.0001;

epsPri_added_term_frst = 0;
epsPri_added_term_scnd = 0;

r = 0;
```

```
    s = 0;

    for i = 1:T

        if i == T

            r = r + norm(r_temp_1(:, :, i), 'fro');
            s = s + rho * norm(s_temp_1(:, :, i), 'fro');

        else

            r = r + norm(r_temp_1(:, :, i), 'fro') + norm(r_temp_2(:, :, i), 'fro') + norm(r_temp_3(:, :, i), 'fro');
            s = s + rho * ( norm(s_temp_1(:, :, i), 'fro') + norm(s_temp_2(:, :, i), 'fro') + norm(s_temp_3(:, :, i), 'fro') );

        end

        epsPri_added_term_frst = epsPri_added_term_frst + epsRel * norm(Theta(:, :, i), 'fro');
        epsPri_added_term_scnd = epsPri_added_term_scnd + epsRel * ( norm(Z0(:, :, i), 'fro') + norm(Z1(:, :, i), 'fro') + norm(Z2(:, :, i), 'fro') );

        epsDual = epsDual + epsRel * rho * ( norm(U0(:, :, i), 'fro') + norm(U1(:, :, i), 'fro') + norm(U2(:, :, i), 'fro') );

    end

    epsPri = epsPri + max(epsPri_added_term_frst,  epsPri_added_term_scnd);


end
```

Plotting the result

```
dev_Theta = zeros(1, T-1);

for i = 1:T-1

    dev_Theta(i) = norm((Theta(:, :, i+1) - Theta(:, :, i)), 'fro');

end

smpld_timstmps = 1:T-1 ;

figure
semilogy(smpld_timstmps, dev_Theta(smpld_timstmps))
hold on
semilogy(smpld_timstmps, dev_Theta(smpld_timstmps), '.','MarkerSize', 12)
% axis square
axis tight
```

```
ylim([1e-3, 1])
set(gca,'xtick',2:5:12,'xticklabel',{'Jan','Feb','Mar'})
ylabel('Temporal Deviation')
title('Perturbed Node Detection for Finance Data')


toc;
```

References

D. Hallac, Y. Park, S. Boyd, J. Leskovec. Network Inference via the Time-Varying Graphical Lasso In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017

A. Ahmed and E. P. Xing. Recovering time-varying networks of dependencies in social and biological studies. PNAS, 2009.

O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. JMLR, 2008.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine learning, 2011.

S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

J. Dahl and L. Vandenberghe. CVXOPT: A Python package for convex optimization. In Proc. Eur. Conf. Op. Res, 2006.

P. Danaher, P. Wang, and D. Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. JRSS: Series B, 2014.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 2008.

M. Gomez Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In KDD. ACM, 2010.

M. S. Grewal. Kalman filtering. Springer, 2011.

D. Hallac, J. Leskovec, and S. Boyd. Network lasso: Clustering and optimization in large graphs. In KDD, 2015.

D. Hallac, C. Wong, S. Diamond, R. Sosi?, S. Boyd, and J. Leskovec. SnapVX: A network-based convex optimization solver. JMLR (To Appear), 2017.

T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning. Springer, 2009.

M. R. Hestenes. Multiplier and gradient methods. Journal of Optimization Theory and Applications, 1969.

C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack. BIG & QUIC: Sparse inverse covariance estimation for a million variables. In NIPS, 2013.

M. Kolar, L. Song, A. Ahmed, and E. Xing. Estimating time-varying networks. The Annals of Applied Statistics, 2010.

D. Koller and N. Friedman. Probabilistic Graphical Models: Principles and Techniques. MIT press, 2009.

S. L. Lauritzen. Graphical models. Clarendon Press, 1996.

K. Mohan, M. Chung, S. Han, D. Witten, S.-I. Lee, and M. Fazel. Structured learning of Gaussian graphical models. In NIPS, 2012.

K. Mohan, P. London, M. Fazel, D. Witten, and S.-I. Lee. Node-based learning of multiple Gaussian graphical models. JMLR, 2014.

P. C. Molenaar. A dynamic factor model for the analysis of multivariate time series. Psychometrika, 1985.

R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana. Estimating time-varying brain connectivity networks from functional MRI time series. Neuroimage, 2014.

S. Myers and J. Leskovec. On the convexity of latent social network inference. In NIPS, 2010.

A. Namaki, A. Shirazi, R. Raei, and G. Jafari. Network analysis of a financial market based on genuine correlation and threshold method. Physica A: Stat. Mech. Apps., 2011.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. Journal of Optimization Theory and Applications, 2016.

N. Parikh and S. Boyd. Proximal algorithms. Foundations and Trends in Optimization, 2014.

H. Rue and L. Held. Gaussian Markov Random Fields: Theory and Applications. CRC Press, 2005.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In NIPS, 2010.

L. Vanderberghe. Proximal mapping lecture notes. http://seas.ucla.edu/ ~vandenbe/236C/lectures/proxop.pdf, 2010.

M. J. Wainwright and M. I. Jordan. Log-determinant relaxation for approximate inference in discrete Markov random fields. IEEE Tr. on Signal Processing, 2006.

K. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In NIPS, 2006.

E. Wit and A. Abbruzzo. Inferring slowly-changing dynamic gene-regulatory networks. BMC Bioinformatics, 2015.

D. Witten and R. Tibshirani. Covariance-regularized regression and classification for high dimensional problems. JRSS: Series B, 2009.

M. Wytock and J. Z. Kolter. Sparse Gaussian conditional random fields: Algorithms, theory, and application to energy forecasting. ICML, 2013.

S. Yang, Z. Lu, X. Shen, P.Wonka, and J. Ye. Fused multiple graphical lasso. SIAM, 2015.

M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. Biometrika, 2007.

S. Zhou, J. Lafferty, and L.Wasserman. Time varying undirected graphs. Machine Learning, 2010.