Learning Sparse Gaussian Graphical Model with l_0-regularization

Beipeng Mu* Jonathan P. How†
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 12139

Abstract

For the problem of learning sparse Gaussian graphical models, it is desirable to obtain both sparse structures as well as good parameter estimates. Classical techniques, such as optimizing the l_1-regularized maximum likelihood or Chow-Liu algorithm, either focus on parameter estimation or constrain to specific structure. This paper proposes an alternative that is based on l_0-regularized maximum likelihood and employs a greedy algorithm to solve the optimization problem. We show that, when the graph is acyclic, the greedy solution finds the optimal acyclic graph. We also show it can update the parameters in constant time when connecting two sub-components, thus work efficiently on sparse graphs. Empirical results are provided to demonstrate this new algorithm can learn sparse structures with cycles efficiently and that it dominates l_1-regularized approach on graph likelihood.

1 Introduction

Graphical models are compact probabilistic representations of the conditional dependence of random variables [1–4]. They have been widely used in probability theory, Bayesian statistics, machine learning, and successful applications include speech recognition, computer vision, social networks, genetics, and diagnosis of diseases [5–9]. When the dependence between variables is bidirectional and the distribution of variables are Gaussian, the graphical model is also called Gaussian graphical models (GGM). This paper discusses the learning of sparse GGMs. Sparse GGMs are desirable in that they can represent variable correlations with fewer parameters, thereby leading to more efficient storage and faster inference [10–12]. Learning a GGM has two aspects: learning the structure, i.e.,

*mubp@mit.edu
†jhow@mit.edu

how variables are connected to each other; and learning parameters, i.e., the correlations between connected variables.

When the graph is assumed to be a tree, exact maximal likelihood (ML) estimates of the structure can be obtained by the Chow-Liu algorithm [13]. However, the representational capability of trees is limited and thus not applicable to many problems of interest, such as social networks, gene interactions, sensor networks and so on. Several models have been proposed to extend that algorithm into non-tree graphs [14–17]. In particular, the recent work of Liu et al. [17] presents a Conditional Chow-Liu algorithm that divides the variables into tree variables and feedback variables. Conditioned on the feedback variables, the tree variables form a tree, thus can be learned by the Chow-Liu algorithm. The feedback variables are fully connected to each other and to tree variables. Thus, when the feedback variables are not known a priori, an approximate algorithm is given to select them in a greed fashion, and the number of edges grows quickly with number of feedback variables. Since Chow-Liu like algorithms are based on trees, they cannot deal well with unconnected graphs. Furthermore, they mainly focus on learning the structure of a graph and therefore require extra efforts to optimize the parameters, which can be expensive in large non-tree graphs. For example, every time a new feedback variable is to be selected, the algorithm computes the marginal gain of all candidates and picks the best one, which involves evaluating parameters of the entire graph, and can be costly when the graph is large [17].

Other researchers have investigated techniques that maximize the likelihood by directly optimizing parameters in the information matrix [18–24]. To avoid over-fitting and achieve sparsity, l_1-regularization was applied in these models. l_1-regularized ML is convex, so classical nonlinear optimization methods can be applied to optimize the cost function. For example, line search in GLASSO [21], interior point in IPM [23], and quasi-Newton in QUIC [24]. The parameter estimate of QUIC converges to the optimal l_1-regularized parameters with quadratic speed and can easily scale to large problems. However, these models do not have explicit mechanics to control sparsity and purely rely on l_1-regularization to have sparse structures. In reality, l_1 does not necessarily give the desired sparse graph for good estimate of the parameters. To obtain the desired sparse graph, the l_1 penalty must be made quite large, thereby distorting the parameters far from ML estimate.

This paper focuses on constructing sparse GGMs that also give good ML estimates of the parameters. This goal is achieved by using a l_0-regularized ML cost function, which is then reduced to a mixed integer programming (MIP) problem. However, the l_0 norm is not convex, MIPs are NP-hard, so exact solutions are intractable. To obtain a tractable solution, a greedy algorithm is presented to select edges sequentially. It is shown that, when the new edge links two unconnected components of the graph, the parameter update can be achieved in constant time, thus the greedy algorithm scales well on sparse graphs where the number of cycles is small compared to the total number of edges. Furthermore, the greedy algorithm reduces to Chow-Liu algorithm (which is optimal) over tree-structured graphs. Empirical results show that the proposed algorithm can

correctly identify sub-components and cycles in them rather than an all connected acyclic graph by Chow-Liu algorithm. It dominates the likelihood of l_1-regularized cost function with same level of sparsity.

Section 2 sets up notations and formulates the problem as a Mixed Integer Programming (MIP) problem. Section 3 presents the approximate algorithm to solve the MIP problem and related theoretical propositions. Section 4 numerically compares the proposed algorithm with previously presented l_1-regularized method QUIC [24] and conditional Chow-Liu [17].

2 Problem Setup

Let $X = \{x^1, \cdots, x^n\}$ be an n-dimensional random vector following an n-variate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, with mean μ and covariance Σ. Denote $P = \Sigma^{-1}$ as inverse of the covariance matrix, which is also called the *precision* or *information* matrix. Let $G = (X, E)$ be the corresponding graphical model. E is the set of edges, and $(i, j) \in E$ if and only if x^i and x^j are not independent given all the remaining variables. It can be shown that in GGM, a non-zero entity in P corresponds to an edge in E, therefore the GGM structure is equivalent zero patterns of P, and the GGM parameters are equivalent to learning the non-zero entities of P (see [20]).

Given N samples $x_1, \cdots, x_N$, define S to be the sample covariance matrix, $S = \frac{1}{n}\sum_{i=1}^{N}(x_i - \hat{\mu})(x_i - \hat{\mu})^T$, where $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$. Given an estimate P, the sample log likelihood l is:

$$l(P) = NJ(P), \qquad J(P) = \log|P| - \mathrm{tr}(SP), \tag{1}$$

where $\mathrm{tr}(\cdot)$ denotes trace of a matrix and $|\cdot|$ denotes determinant. Maximizing likelihood is equivalent to maximizing $J(P)$. Without any regularization, $P_{ML} = \mathrm{argmax}_P\, J(P) = S^{-1}$, which is typically not sparse. l_1-regularization is a widely used option [19, 20, 24] to increase the sparsity:

$$\max_{P \succ 0} J_1(P) \quad = \quad \min_{P \succ 0} \ \mathrm{tr}(SP) - \log|P| + \lambda\|P\|_1. \tag{2}$$

The l_1-regularized estimates are empirically more sparse than that of the ML estimate[20], but this is accomplished at the expense of distorting P from the desired parameter estimates P_{ML}.

To get desired sparsity level but maintain the high likelihood of the estimate P, we instead use l_0-regularization:

$$\max_{P \succ 0} J_0(P) \quad = \quad \min_{P \succ 0} \ \mathrm{tr}(SP) - \log|P| + \lambda\|P\|_0. \tag{3}$$

l_0 norm can be computed as the number of non-zero entities in the matrix P. Also note that the cost function (3) is the Akaike Information Criteria[25] applied in GGM learning. AIC has been widely used in evaluating models. For the same structure, l_0 norm is constant over parameters in P, so $J_0(P)$ is maximal when P is the maximal likelihood estimate.

Define binary variables z_{ij} as indicators whether edge (i, j) is selected in the graph. Denote Z as the set of selected edges, $Z = \{(i, j)|z_{ij} = 1\}$, and $\bar{Z}$ as the complementary set of Z. Denote P_Z as the set of elements at position Z. Then (2) is equivalent to:

$$\begin{aligned} \max_{P,Z} J_0(P, Z) = \min_{P,Z,M} \operatorname{tr}(SP) - \log\det(P) + \lambda M \qquad (4) \\ \text{s.t.} \quad P_{\bar{Z}} = 0 \\ \sum_{i,j} z_{ij} \leq M \\ z_{ij} \in \{0, 1\}, \quad P \succ 0 \end{aligned}$$

where M is the total number of edges. Because $P \succ 0$ is semi-definite and symmetric, the binary variables that matter are $\{z_{ij}|i < j\}$. There are $\frac{n(n-1)}{2}$ of them to be decided.

3 Solve l_0-regularized ML estimate

Problem (4) is a mixed integer (MIP) semidefinite optimization problem. An exact solution can be obtained by enumerating Z, then optimizing P_Z given Z. However, the complexity of this approach is combinational and thus intractable for large problems. Instead of solving the exact problem, we select the edge in a greedy manner, similar to how [17] greedily selects the feedback variables, but in this approach we can directly control the sparsity and number of edges.

3.1 Edge Selection Criteria

Define $f(Z)$ as the optimal cost under graph structure Z:

$$f(Z) = \max_{P_{\bar{Z}}=0} J_0(P, Z) \qquad (5)$$

then the optimal solution can be written as $\max_Z f(Z)$. An intuitive way of selecting edges would be to start from no edges, and then greedily add the edge that gives the maximal increment change in $f(Z)$:

$$(i, j) = \arg \max_{(i,j)\in\bar{Z}} f(Z \cup (i, j)) - f(Z).$$

With this heuristic, each time there is a new edge is added, the algorithm must recompute $f(Z \cup (i, j))$ for all $(i, j) \in \bar{Z}$. Notice $f(Z \cup (i, j))$ is defined as the optimal P given structure Z, so computing $f(Z)$ would involve optimizing P under the structure Z, which can be slow for large networks.

The alternative approach taken here is to pick the edge that has the steepest gradient:

$$(i, j)^* = \operatorname*{argmax}_{(i,j)\in\bar{Z}, i<j} \left\| \frac{dJ_0(P, Z)}{d(P)_{ij}} \right\|_1 = \operatorname*{argmax}_{(i,j)\in\bar{Z}, i<j} \|(S)_{ij} - (P^{-1})_{ij}\|_1 \qquad (6)$$

Algorithm 1 l_0-regularized GMM estimate

1: **Input**: empirical covariance S, number of edges M
2: **Output**: information matrix P
3: $Z_0 = \{(i,i)\}, \quad i = 1, \cdots, n$
4: $(P_0)_{ii} = 1/(S)_{ii}, \quad i = 1 \cdots, n$
5: **for** $m = 1 : M$ **do**
6: Select a new edge
7: $(i_m^*, j_m^*) = \text{argmax}_{(i,j)\in\bar{Z}_k, i<j} |(S)_{ij} - (P_{m-1}^{-1})_{ij}|_1$
8: Update structure and parameters
9: $Z_m = Z_{m-1} \cup (i_m^*, j_m^*) \cup (i_m^*, j_m^*)$,
10: $P_m = f(Z_m)$
11: **end for**

where $\|\cdot\|_1$ is equivalent to taking absolute value. This idea is similar to that in l_1-regularized approaches [24] that only edges with have steep gradients are active.

3.2 Stopping Criteria and Alternative MIP

Every time a new edge (i,j) is added, the l_0 penalty in cost function (4) increases by 2λ (edge (j,i) is also added because of symmetry), and the likelihood $J(P)$ increases by $f(Z \cup (i,j)) - f(Z)$. Then the intuitive stopping criteria is when the incremental change in f does not exceed the regularization penalty, i.e., $f(Z \cup (i,j)) - f(Z) < 2\lambda$. With this criteria, λ controls the sparsity of the learned graph, and maps uniquely into a M. Instead of having λ as an input variable and optimizing over M, an equivalent way is to directly have M as an input variable, and (4) becomes:

$$\begin{aligned} \min_{P,Z} J_0(P,Z) = \min_{P,Z} \ & \text{tr}\,(SP) - \log|P| \\ \text{s.t.} \quad & P_{\bar{Z}} = 0 \\ & \sum_{i,j} z_{ij} \le M \\ & z_{ij} \in \{0,1\}, \quad P \succ 0 \end{aligned} \tag{7}$$

Combining the edge selection criterion and the stopping criteria, the l_0-regularized approach is presented in Algorithm 1.

The most computation consuming steps in Algorithm 1 are computing the matrix inverse $(P_{m-1})_{ij}^{-1}$ in line 7 and updating parameters $P_m = f(Z_m)$ in line 10. The following propositions shows this procedure can be done efficiently when the new edge connects two sub-components, which would enable the algorithm to scale well on large sparse graphs.

Proposition 1 If edge (i_m^*, j_m^*) is added into the graph at step m and j_m^*

are not connected in P_{m-1}, then P_m can be computed analytically in $\mathcal{O}(1)$:

$$\begin{aligned} P_m =& P_{m-1} + \begin{pmatrix} e_{i_m^*} & e_{j_m^*} \end{pmatrix} \begin{pmatrix} r & x \\ x & q \end{pmatrix} \begin{pmatrix} e_{i_m^*}^T \\ e_{j_m^*}^T \end{pmatrix} \\ d =& S_{i_m^* i_m^*} S_{j_m^* j_m^*} - S_{i_m^* j_m^*}^2 \\ r =& \frac{S_{i_m^* j_m^*}^2}{S_{j_m^* j_m^*} d}, \qquad q = \frac{S_{i_m^* j_m^*}^2}{S_{i_m^* i_m^*} d}, \qquad x = \frac{S_{i_m^* j_m^*}}{d} \end{aligned} \tag{8}$$

where $e_i = (0, \cdots, 0, 1, 0, \cdots, 0)^T$ is unit vector with appropriate length.

Proof 1 For brevity, in the proof we leave out superscripts and subscripts without causing confusion, write (i_m^*, j_m^*) as (i, j). If i and j are not connected in P_{m-1}, then P_{m-1} can be rearranged into a block matrix with zero off-diagonal entities:

$$P_{m-1} = \begin{pmatrix} Q_{m-1} & 0 \\ 0 & R_{m-1} \end{pmatrix} \tag{9}$$

where Q_{m-1} is the part of the graph containing variable x_i, R_{m-1} is the part of the graph containing variable x_j, and Q_{m-1}, R_{m-1} are not connected. Given Z_{m-1}, $J_0(P, Z_{m-1})$ (defined in (7)) is differentiable over P, so $P_{m-1} = \operatorname{argmin} J_0(P, Z_{m-1})$ must satisfy the first order necessary condition:

$$\left(\frac{dJ_0(P_{m-1}, Z_{m-1})}{dP_{m-1}} \right)_{Z_{m-1}} = \left(S - P_{m-1}^{-1} \right)_{Z_{m-1}} = 0 \tag{10}$$

where $()_{Z_{m-1}}$ represent the matrix elements in location Z_{m-1}. Denote S_Q as the sub-matrix of S corresponds to the positions of Q_{m-1} and so for S_R, plug into (10):

$$(S_Q - Q_{m-1}^{-1})_{Z_{m-1}} = 0, \quad (S_R - R_{m-1}^{-1})_{Z_{m-1}} = 0 \tag{11}$$

After adding an edge (i, j) with parameter x, P_m has the form:

$$P_m = \begin{pmatrix} Q_m & e_i e_j^T x \\ e_j e_i^T x & R_m \end{pmatrix} \tag{12}$$

The first order necessary condition for P_m is:

$$\begin{pmatrix} S_Q & e_i e_j^T S_{ij} \\ e_j e_i^T S_{ij} & S_R \end{pmatrix}_{Z_m} - \begin{pmatrix} Q_m & e_i e_j^T x \\ e_j e_i^T x & R_m \end{pmatrix}_{Z_m}^{-1} = 0 \tag{13}$$

Applying the following theorem of inverting block matrix,

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix}^{-1} = \begin{pmatrix} (A - BC^{-1}B^T)^{-1} & -A^{-1}B(C - B^T A^{-1} B)^{-1} \\ -(C - B^T A^{-1} B)^{-1} B^T A^{-1} & (C - B^T A^{-1} B)^{-1} \end{pmatrix} \tag{14}$$

$$\begin{pmatrix} Q_m & e_i e_j^T x \\ e_j e_i^T x & R_m \end{pmatrix}^{-1} = \begin{pmatrix} (Q_m - (R_m^{-1})_{jj} x^2 e_i e_i^T)^{-1} & g \\ g^T & (R_m - (Q_m^{-1})_{ii} x^2 e_j e_j^T)^{-1} \end{pmatrix}$$
$$g = - Q_m^{-1} e_i e_j^T x (R_m - (Q_m^{-1})_{ii} x^2 e_j e_j^T)^{-1} \tag{15}$$

Plug (15) into (13) and take the block-wise equality:

$$\begin{aligned} (S_Q)_{Z_m} =& ((Q_m - (R_m^{-1})_{jj} x^2 e_i e_i^T)^{-1})_{Z_m} \\ (S_R)_{Z_m} =& ((R_m - (Q^-1_m)_{ii} x^2 e_j e_j^T)^{-1})_{Z_m} \\ e_i e_j^T S_{ij} =& g \end{aligned} \tag{16}$$

$Z_m = Z_{m-1} \cup (i,j)$, so $Z_m = Z_{m-1}$ at block Q and R. Further plug in (11):

$$\begin{aligned} ((Q_m - (R_m^{-1})_{jj} x^2 e_i e_i^T)^{-1})_{Z_m} =& (S_Q)_{Z_m} = (S_Q)_{Z_{m-1}} = (Q_{m-1}^{-1})_{Z_{m-1}} \\ ((R_m - (Q_m^{-1})_{ii} x^2 e_j e_j^T)^{-1})_{Z_m} =& (S_R)_{Z_m} = (S_R)_{Z_{m-1}} = (R_{m-1}^{-1})_{Z_{m-1}} \\ S_{ij} =& e_i^T g e_j = -(Q_m^{-1})_{ii} S_{jj} x \end{aligned} \tag{17}$$

It is sufficient to satisfy (17) by only updating 4 entities of P_{m-1}: (i,j), (i,j), (j,i) and (j,j).

$$\begin{aligned} Q_m =& Q_{m-1} + (R_m^{-1})_{jj} x^2 e_i e_i^T \\ R_m =& R_{m-1} + (Q_m^{-1})_{ii} x^2 e_j e_j^T \\ S_{ij} =& - (Q_m^{-1})_{ii} S_{jj} x \end{aligned} \tag{18}$$

Define $q = (Q_m)_{ii} - (Q_{m-1})_{ii}$, $r = (R_m)_{ii} - (R_{m-1})_{ii}$ and plug in (18)

$$\begin{aligned} q =& (R_m^{-1})_{jj} x^2 = ((R_{m-1} + e_i e_i^T r)^{-1})_{jj} x^2 \\ r =& (Q_m^{-1})_{ii} x^2 = ((Q_{m-1} + e_i e_i^T q)^{-1})_{ii} x^2 \\ S_{ij} =& - (Q_m^{-1})_{ii} S_{jj} x = -((Q_{m-1} + e_i e_i^T q)^{-1})_{ii} S_{jj} x \end{aligned} \tag{19}$$

Apply theorem $(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}$. And recall $(Q_{m-1}^{-1})_{ii} = S_{ii}$, $(R_{m-1}^{-1})_{jj} = S_{jj}$ from (11):

$$q = (R_m^{-1})_{jj} x^2 = \frac{S_{jj} x^2}{1 + r S_{jj}} \qquad r = (Q_m^{-1})_{ii} x^2 = \frac{S_{ii} x^2}{1 + q S_{ii}} \qquad \frac{S_{ii} S_{jj}}{1 + q S_{ii}} + S_{ij} = 0 \tag{20}$$

Solve these equation and get (8).

Compared to P_{m-1}^{-1}, P_m^{-1} is different only in row and column i_m^*, j_m^*, thus can be computed with $\mathcal{O}(N^2)$ complexity instead of $\mathcal{O}(N^3)$. When the new selected edge connects two variables that are already in the same component, new cycles within this component will be formed, and all parameters in the component will change. In this case, standard convex optimization algorithms

can be used to update the parameters. Given Z, $J(P, Z)$ is twice differentiable over P, the optimization can be done at least at quadratic convergence speed. On sparse graphs, we expect this kind of parameter updates do not happen a lot.

The next proposition shows the proposed approach reduces to Chow-Liu algorithm on acyclic trees. Because Chow-Liu tree learns the optimal acyclic graph, it obtains optimality on acyclic graphs.

Proposition 2 Assume that the variables are zero-mean, unit-variance(If not, variables can be normalized first). If the graph learned by algorithm 1 is acyclic, it is the optimal acyclic graph.

Proof 2 The approach is to prove that the procedure in algorithm 1 is equivalent to the Chow-Liu procedure when the learned graph is a tree. First note that because the variables are normalized, the empirical covariance matrix S is a correlation matrix: $S_{ij} = \mathbf{corr}(x_i, x_j)$, and $\|S_{ij}\|_1 \leq 1$.

Chow-Liu algorithm greedily selects edges that does not form cycles based on mutual information. The mutual information of two Gaussian variables x_i and x_j conditioned on all other variables is:

$$I_{ij} = -\frac{1}{2} \ln \left(1 - \text{corr}(x_i, x_j)^2\right) \tag{21}$$

At step m, define the set of edges that keep the graph acyclic as $\tilde{S}$, and $\tilde{S} \subset \bar{S}$. When the learned graph is acyclic, $(i_m, j_m)^* \in \tilde{S}$, and $(P_{m-1}^{-1})_{i_m^*, j_m^*} = 0$. From the edge selection criteria (6), we have:

$$\begin{aligned}
(i_m, j_m)^* &= \underset{(i,j)\in\bar{S}}{\text{argmax}} \|S_{ij} - (P_{m-1}^{-1})_{ij}\|_1 = \underset{(i,j)\in\tilde{S}}{\text{argmax}} \|S_{ij} - (P_{m-1}^{-1})_{ij}\|_1 \\
&= \underset{(i,j)\in\tilde{S}}{\text{argmax}} \|S_{ij}\|_1 = \underset{(i,j)\in\tilde{S}}{\text{argmax}} \|\text{corr}(x_i, x_j)\|_1 = \underset{(i,j)\in\tilde{S}}{\text{argmin}} (1 - \text{corr}(x_i, x_j)^2) \\
&= \underset{(i,j)\in\tilde{S}}{\text{argmax}} -\frac{1}{2} \ln(1 - (x_i, x_j)^2) = \underset{(i,j)\in\tilde{S}}{\text{argmax}} I_{i,j}
\end{aligned} \tag{22}$$

The last step is exactly the Chow-Liu criterion, so algorithm 1 fully recovers the Chow-Liu edge selection procedure and thus returns the optimal acyclic graph.

4 Experiment

This section compares the l_0-regularized method with a state-of-the-art l_1-regularized method QUIC [24], and a state-of-the-art Chow-Liu based method called Conditioned Chow-Liu (CCL) [17] on a synthetic dataset as well as an online wine dataset.

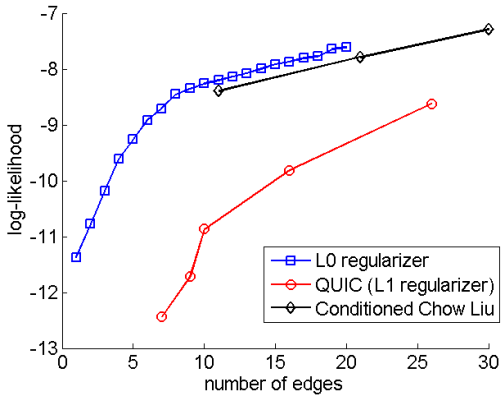
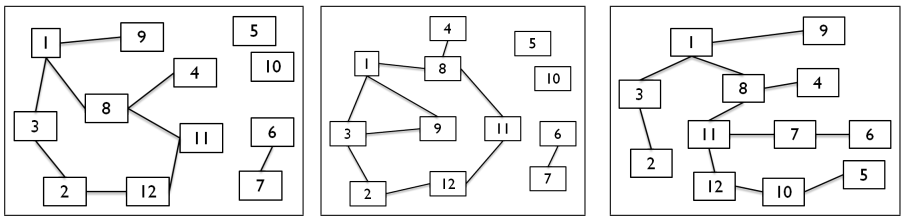


Figure 1: Comparison of log likelihood over sparsity

4.1 Wine Dataset

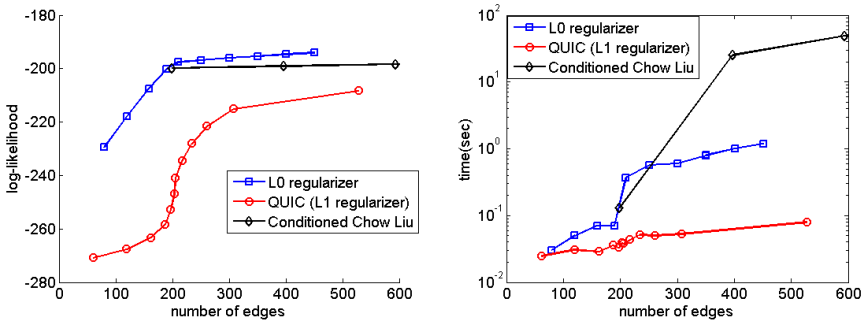
The dataset obtained from UCI on-line repository for machine learning tests[**?**]. This dataset is a chemical analysis of wines grown in the same region in Italy but derived from different cultivars. The analysis determined the quantities of 12 attributes found in 1599 samples of wines. These 12 attributes are: 1) fixed acidity, 2) volatile acidity, 3) citric acid, 4) residual sugar, 5) chlorides, 6) free sulfur dioxide, 7) total sulfur dioxide, 8) density, 9) pH, 10) sulphates, 11) alcohol, 12) quality, The data is first normalized by subtracting its mean and divided by the standard deviation. Then the l_0-regularized, l_1-regularized (QUIC) and Conditional Chow-Liu(CCL) algorithms were performed to learn the correlation of these attributes.

Figure 1 gives the log likelihood $J(P)$ of the different methods. As shown, the l_0-regularized method and CCL obtain good parameter estimates. More specifically, the l_0-regularized method has much higher likelihood than that of QUIC given the same level of sparsity and the Conditional Chow-Liu is just slightly worse. Figure 2 shows the learned structures. For comparison, we pick the structures learned by QUIC and l_0 method that have the same number of edges. Figure 2 (c) shows the CL tree. It can be seen that QUIC and the l_0-regularized method are able to detect cycles in the graph. CCL only add edges on top of CL tree, so it can only be denser than a tree. Every time a feedback node is selected, it connects to all other non-feedback nodes, thus the number of edges can only be multiples of n. Also note that the slope change of the log likelihood of the l_0-regularized method after 9 edges, with a similar feature in QUIC at 10 features, which suggest that, for this case, 9–10 features would be a good balance between likelihood and complexity of the graph.



(a) l_0 regularized (b) QUIC (c) CCL

Figure 2: Comparison of graph structure



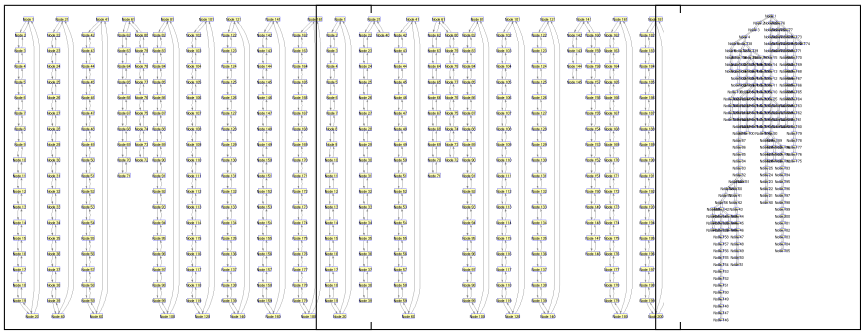
(a) Log likelihood (b) Running time

Figure 3: Comparison on synthetic dataset

4.2 Synthetic Data

In this section, we create a synthetic dataset whose corresponding graph is sparse and cyclic. In this synthetic dataset, there are 200 variables, and they form 10 twenty-variable sub-components. In each component the variables are connected in a circle, the diagonal elements of each sub-component are set to be 1.25. To maintain positive definiteness, the off-diagonal elements of each sub-component are set to be -0.5. A total of 200 Gaussian samples with a mean of zero were generated from this constructed graph to obtain the empirical covariance matrix.

Figure 3a shows the log likelihood of the same three approaches considered in example 1. As before, the log likelihood of the l_0-regularized approach dominates that of QUIC. The l_0-regularized approach is also better than CCL in that it has higher likelihood. The l_0-regularized result plateaus after 200 edges, which is the underlying truth. Therefore l_0 regularization also provides a way to infer the underlying true structure. Figure 3b shows the total run time of the three approaches. For sparse structure (less than 200 edges), l_0-regularized approach has similar run time as QUIC, as it is able to update most parameters for new selected edges in constant time. When the graph becomes denser, more cycles are formed, l_0-regularized method is slower than QUIC, because it needs to



(a) l_0 regularized (b) QUIC (c) CCL

Figure 4: Structure learned on synthetic data

optimize all parameters within cycles. CCL. is the slowest among the three.

Figure 4 shows the structures learned by 3 approaches. Again l_0-regularized and l_1-regularized approaches are able to handle cycles in graphs, and they correctly identified 10 cycles within the graph. On the other hand, Chow-Liu method can only build a tree, thus connects all the variables together.

5 Conclusion and Future Work

In this paper, we used l_0-regularization to learn sparse GGM model, and developed a greedy algorithm to solve the equivalent mixed integer programming problem. The new proposed algorithm is compared with state of art l_1-regularized method QUIC and tree-based method Conditional Chow-Liu on a wine dataset and synthetic dataset. The numerical results show that l_0-regularization can get the highest likelihood, dominates the l_1-regularized method. It can also correctly identify sub-components in sparse graphs. The likelihood plateaus when the algorithm hits the correct number of edges thus also gives insights in model selection.

Future work include further optimize the algorithm, such as optimizing parameter only over sub-components, intelligent ways to group edges together when doing line search over parameters in cycles, and better heuristic for initializing optimization. We are also interested in testing it on big real world datasets.

References

[1] R. Diestel. *Graph Theory.* Springer, Berlin, DE, second edition, 2000.

[2] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, January 2008.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, first edition, 2007.

[4] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[5] Michael Isard. Pampas: Real-valued graphical models for computer vision. In *Computer Vision and Pattern Recognition, 2003 IEEE Computer Society Conference on*, volume 1, pages I–613. IEEE, 2003.

[6] S. A. Aldosari and J. M. F. Moura. Distributed detection in sensor networks: connectivity graph and small-world networks. In *39th Asilomar Conference on Signals, Systems, and Computers*, pages 230 – 234, Pacific Grove, CA, Oct. 2005.

[7] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[8] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 807–816. ACM, 2009.

[9] Steffen L Lauritzen and Nuala A Sheehan. Graphical models for genetic analyses. *Statistical Science*, pages 489–514, 2003.

[10] Mark Andrew Paskin. *Exploiting locality in probabilistic inference*. PhD thesis, Berkeley, CA, USA, 2004. AAI3165519.

[11] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

[12] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *AAAI*, pages 133–136, 1982.

[13] C.K. Chow and C.N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans Info Theory*, IT-14(3):462–467, May 1968.

[14] David Karger and Nathan Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401, Philadelphia, PA, USA, 2001.

[15] M. J. Choi, V. Chandrasekaran, and A.S. Willsky. Gaussian multiresolution models: Exploiting sparse markov and covariance structure. *Signal Processing, IEEE Transactions on*, 58(3):1012–1024, March 2010.

[16] Pieter Abbeel, Daphne Koller, and Andrew Y. Ng. Learning factor graphs in polynomial time and sample complexity. *J. Mach. Learn. Res.*, 7:1743–1788, December 2006.

[17] Ying Liu and Alan Willsky. Learning gaussian graphical models with observed or latent fvss. In *Advances in Neural Information Processing Systems 26*, pages 1833–1841. 2013.

[18] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.*, 9:485–516, June 2008.

[19] K Scheinberg, S Ma, and D Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2101–2109. 2010.

[20] Figen Oztoprak, Jorge Nocedal, Steven Rennie, and Peder A. Olsen. Newton-like methods for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 764–772. 2012.

[21] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9, 2009.

[22] John Duchi, Stephen Gould, and Daphne Koller. Projected subgradient methods for learning sparse gaussians. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

[23] Lu Li and Kim-Chuan Toh. An inexact interior point method for l1-regularized sparse covariance selection. *MATHEMATICAL PROGRAMMING COMPUTATION*, 2, 2010.

[24] Cho-Jui Hsieh, Matyas A. Sustik, Inderjit S. Dhillon, and Pradeep Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems 24*, pages 2330–2338. 2011.

[25] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, Dec 1974.