ECE 830
Note on Graphical Models

1 Introduction

We have focused mainly on linear models for signals, in particular the subspace model $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{\theta}$, where $\boldsymbol{H}$ is a $n \times k$ matrix and $\boldsymbol{\theta} \in \mathbb{R}^k$ is a vector of $k < n$ parameters describing the signal $\boldsymbol{x}$. The subspace model is useful because it reduces the number of parameters or degrees of freedom in the model from n to k. While applicable to many real-world problems, this is not the only way of modeling signals with a small number of parameters.

Another widely used approach is called graphical modeling. The basic idea in a graphical model is treat the variables in the signal vector $\boldsymbol{x}$ as random variables and explicitly represent probabilistic relationships between the variables. More specifically, each of the n variables is represented as a vertex in a graph. Probabilistic relationships between variables are edges in the graph. If two variables are conditionally independent (more on this in a moment), then there will be no edge between them. In general, a graph with n vertices can have up to $O(n^2)$ edges. Each edge can be viewed as a degree of freedom in the graphical model. If the number of edges is limited to a smaller number, then we have a model with fewer degrees of freedom. Graphical models are also often referred to as *Bayesian Networks*.

Consider the example graph shown in Figure 1 below. The graphical model represents a joint distribution $p(x_1, x_2, \ldots, x_7)$. Specifically, the edges indicate constraints that the joint distribution $p(x_1, x_2, \ldots, x_7)$ satisfies. If two variables are conditionally independent given all other variables, then this is indicated by the absence of an edge between the two variables. For example, the graph structure tells us that x_2 and x_2 are conditionally independent given all other variables; i.e.,

$$p(x_2, x_3|x_1, x_4, x_5, x_6, x_7) \;=\; p(x_2|x_1, x_4, x_5, x_6, x_7)p(x_3|x_1, x_4, x_5, x_6, x_7),$$

the conditional distribution of x_2 and x_3 factorizes. The graph tells us even more. Let $n(i)$ be the vertices connected to vertex i and let $\boldsymbol{x}_{n(i)}$ denote the set of corresponding variables. Then $x_i|\boldsymbol{x}_{n(i)}$ is (conditionally) independent of all other variables. For example, $p(x_1, x_4, \ldots, x_7|x_2, x_3) = p(x_1|x_2, x_3)p(x_4, \ldots, x_7|x_2, x_3)$ and $x_4|x_2$ is independent of all other variables.

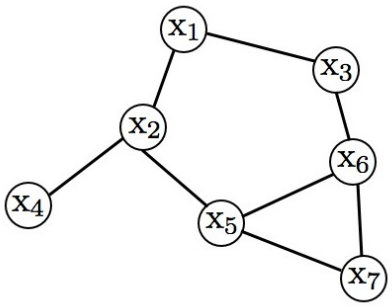

Figure 1: Graphical representation of 7 interdependent random variables.

To give a bit more insight, let's look at a number of very common graphical models, starting off with the most basic.

1.1 Markov Chains

Suppose that $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$. For example, we can imagine that the x_i are samples of a signal in time. If the signal is random, then we talk about the joint probability distribution $p(x_1, \ldots, x_n)$. If the x_i are independent, then $p(x_1, \ldots, x_n) = \prod_{i=1}^n p(x_i)$, where $p(x_i)$ is the marginal distribution for x_i. A (first-order) Markov model puts a bit more structure on the signal by assuming that the x_i are dependent, but *conditionally independent* in the following sense:

$$p(x_1, \ldots, x_n) \quad = \quad p(x_1) \prod_{i=1}^{n-1} p(x_{i+1}|x_i)$$

This sort of factorization of the joint density implies that each x_{i+1} depends on $x_1, \ldots, x_i$, but is conditionally independent of $x_1, \ldots, x_{i-1}$ given x_i. This is precisely the sort of factorization we assumed in the Kalman and particle filtering lectures. The graph associated with this factorization is shown in Fig 2. The edges/arrows in the graphical representation show the conditional independence relationships among the variables. Assuming (full) independence among the variables is a degenerate case of the Markov chain. The graph in that case would have no edges. Markov chain models are used in wireless communications, automatic speech analysis and synthesis, and many other applications.

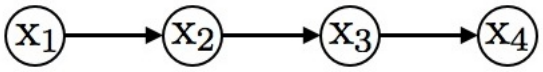

Figure 2: Markov chains are "linear" graphs.

1.2 Trees

Tree-structured graphical models are a simple generalization of the chain. A tree model is depicted in Fig. 3. In a tree, the top vertex (x_1 in the figure) is called the root. The the vertices at the bottom ($x_3, \ldots, x_5$ in the figure) are called leaves. The graph represents the following factorization of the joint density. Each variable is conditionally independent of all the variables above in the tree given the variable directly above it (its "parent"). For example, x_2 is the parent of x_3 and x_4 in the figure. Denote the parent of x_i by $x_{p(i)}$. Then the joint density can be written as

$$p(x_1, \ldots, x_{15}) \quad = \quad p(x_1) \prod_{i=2}^{5} p(x_i|x_{p(i)})$$

Trees are widely-used in image processing and machine learning.

1.3 Directed Acyclic Graphs (DAGS)

DAGs are a further generalization of chains and trees. A DAG is shown in Fig. 4. Notice the red arrows on the edges. These indicate the "direction" of the conditional independence/dependence relations. In the chain and tree cases, the direction was implicit, but for more general graphs we need to explicitly indicate the direction to specify the factorization. Note that this graph, without the arrows, would be a loop (or

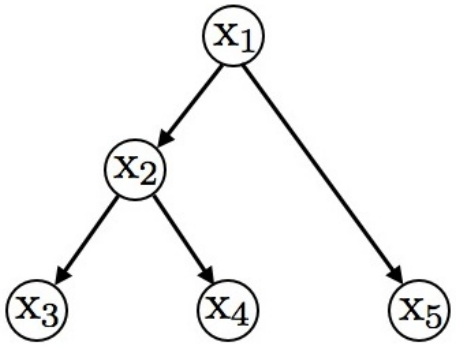

Figure 3: Trees are simple generalizations of linear graphs.

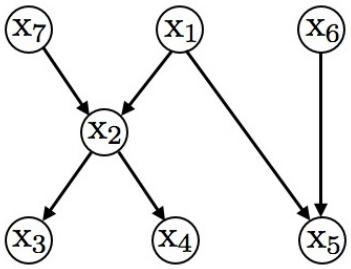

Figure 4: Even more general than chains and trees, DAGs can represent more complex factorizations.

"cycle"), and thus the ordering of the factorization wouldn't be clear. With the arrows, the ordering is clear and the graph does not have a cycle (hence it is an "acyclic" graph). The probability factorization indicated in this figure is

$$p(x_1, \ldots, x_7) \quad = \quad p(x_1)\, p(x_7)\, p(x_6)\, p(x_2|x_1, x_7)\, p(x_3|x_2)\, p(x_4|x_2)\, p(x_5|x_1, x_6) \ .$$

DAGs are used in all sorts of applications, including modeling networks of interactions in biological systems.

1.4 Undirected Graphical Models

Graphical models need not have directed dependencies. A canonical example arises in image processing. Consider an $n \times n$ image modeled as a random process. It is natural to suppose that the value of each pixel depends on neighboring pixels, but not so much on far away pixels. Mathematically, this can be modeled by assuming that each pixel is conditionally independent given the values of its neighbors (e.g., its 4 nearest neighbors). This produces a graphical model that has a lattice structure, as shown in Fig. 5. There is no obvious direction to the dependencies in this model (no notion of causality as there is in temporal modeling). Moreover, the lattice graph contains many cycles. The edges in the graph represent conditional independence

relationships, but because of the cyclic nature of the graph, it is not possible to directly factorize the joint density. This sort of graphical model is often called a Markov random field (MRF). MRFs can be used for statistical inference, but are more difficult to work with than chains, trees, or DAGs.

The joint distribution of the variables in an MRF can be represented in terms of a factorization in terms of *potential functions*, a result known as the Hammersley-Clifford theorm. The factorization isn't as simple or obvious as in the case of DAGs, but it has a similar flavor. To explain this factorization we first need to define the notion of a *clique*. A clique is a fully-connected subgraph of a larger graph. In the MRF in Fig. 5 the cliques are all pairs of (vertically or horizontally) neighboring pixels. Let x_c, $c \in C$, denote the set of

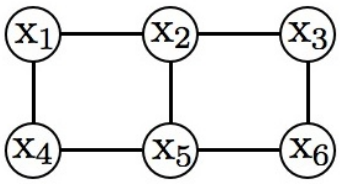

Figure 5: MRFs can model conditional independence relationships without directions. The MRF shown could be appropriate to model the pixels in an 2×3 image.

all cliques for a particular MRF. Then the joint density can be written as

$$p(x_1, \ldots, x_n) \quad = \quad \frac{1}{Z} \prod_{c \in C} \psi_c(x_c)$$

where the ψ denote the potential functions. The form of the potential functions depends on the sort of probabilistic relationships assumed in the MRF. The quantiy Z is called the *partition function*. It is a normalization function need to make the factorization a proper density function and is given by

$$Z \; = \; \int_x \prod_{c \in C} \psi_c(x_c).$$

Herein lies the difficulty of MRFs. Computing the partition function is usually not easy, especially for large graphs like one for an $n \times n$ image. So the factorizating isn't immediately useful as a computational tool. However, over the years many techniques have been developed for approximating the partition function and these lead to reasonable algorithms in practice.

2 Bayesian Inference Based on Graphical Models

Suppose that we observe data $\boldsymbol{y}$ related to a signal of interest $\boldsymbol{x}$ through the likelihood function $p(\boldsymbol{y}|\boldsymbol{x})$. For example, this could correspond to a linear Gaussian observation model: $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}$, $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Suppose that we model our prior knowledge of $\boldsymbol{x}$ using a graphical model $p(\boldsymbol{x})$. We can then compute the posterior distribution $p(\boldsymbol{x}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})$ and use it to make inferences about $\boldsymbol{x}$ (e.g., using the MAP or posterior mean estimator). The Wiener and Kalman filters are two classic examples. The Wiener filter prior is a Gaussian Markov Random Field and the Kalman filter prior is a Gausian Markov Chain.

Computing the Wiener and Kalman filter is relatively straight forward because of Gaussian distribution depends only on means and covariances. However, computation with non-Gaussian graphical models can be more complicated, but managable in certain special cases.

2.1 Computing Posteriors for Chains, Trees and DAGs

Posterior distributions with chain, tree or DAG priors can often be computed exactly with efficient algorithms that exploit the simple Markovian structure of these priors. Consider a simple problem in which the likelihood factorizes $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|x_i)$. This corresponds to the situation in which each observation y_i is conditionally independent given $\boldsymbol{x}$, akin to a simple "signal+independent noise" model. The prior for a chain, tree or DAG also has a factorization of the form

$$p(\boldsymbol{x}) \;=\; \prod_{i=1}^{n} p(x_i|x_{\rho(i)}) \;,$$

where $\rho(i)$ denotes the set of parents (vertices directly preceding and directly connected to x_i in the chain, tree or DAG). For convenience, we enumerate the elements of $\boldsymbol{y}$ and $\boldsymbol{x}$ so that x_i is conditionally independent of $x_{i+1}, x_{i+2}, \ldots, x_n$ given $x_1, x_2, \ldots, x_{i-1}$. In other words $\rho(i) \subset \{x_1, x_2, \ldots, x_{i-1}\}$.

Combining the likelihood and the prior, the posterior has the factorized form

$$p(\boldsymbol{x}|\boldsymbol{y}) \;\propto\; p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) \;=\; \prod_{i=1}^{n} p(y_i|x_i)\, p(x_i|x_{\rho(i)}) \;.$$

The (marginal) posterior distribution for x_i can be computed by "integrating out" or "marginalizing over" all other variables:

$$p(x_i|\boldsymbol{y}) \;\propto\; \int_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n} \prod_{i=1}^{n} p(y_i|x_i)\, p(x_i|x_{\rho(i)})\; dx_1 \cdots dx_{i-1} dx_{i+1} \cdots dx_n.$$

The factorization of the graphical model distribution (and the likelihood function) allows us to compute the multidimensional integration by a simple recursion involving a sequence of one-dimensional integrations[1].

The basic algorithm for computing the posterior operates in a two-pass process over the indices. Before stating the general approach, consider the Markov chain $p(\boldsymbol{x}) = p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1})$. In this case

$$p(\boldsymbol{x}|\boldsymbol{y}) \;\propto\; p(y_1|x_1)\, p(x_1) \prod_{i=2}^{n} p(y_i|x_i)\, p(x_i|x_{i-1}) \;.$$

Let us first compute the posterior distribution of x_1 by "integrating out" $x_2, \ldots, x_n$ (called *marginalization*):

$$p(x_1|\boldsymbol{y}) \quad \propto \quad p(y_1|x_1)\, p(x_1) \int_{x_2,\ldots,x_n} \prod_{i=2}^{n} p(y_i|x_i)\, p(x_i|x_{i-1})\, dx_2 \cdots dx_n$$

Now we can distribute the integration within the product, exploiting the factorization. Beginning with $i = n$

[1] This is true for chains, trees, and "singly connected" DAGs (i.e., undirected graph is a tree). More general DAGs require a similar, but somewhat more complicated, method known as the *junction tree algorithm*.

and $q(x_n) = 1$, for $i = n, n-1, \ldots, 2$ recursively define

$$q(x_{i-1}) \;=\; \int_{x_i} p(y_i|x_i)\, p(x_i|x_{i-1})\, q(x_i)\, dx_i \;.$$

The quantity $q(x_{i-1})$ is considered to be the "message" from vertex i to $i-1$. Then we can write

$$p(x_1|\boldsymbol{y}) \quad \propto \quad p(y_1|x_1)\, p(x_1)\, q(x_1) \;.$$

This style of computation is called "message passing" or "belief propagation." Similar message passing strategies can be used to compute $p(x_i|\boldsymbol{y})$ for $i = 2, \ldots, n$. Notice that this calculation required several 1-dimensional integrations, rather than a full n-dimensional integration. This is the key computational advantage of this message passing procedure. To quantify this improvement. Suppose that the variables take only N distinct values (i.e., they are discrete, rather than continuous, variables) or equivalently suppose that we numerically approximate the integrals by N-term Riemann sums. Then the integrals become sums and $q(x_{i-1})$ involves N terms. We must compute this sum for each of the N values that x_{i-1} can take, and so the total computational complexity required to compute the N values of $q(x_{i-1})$ requires $O(N^2)$ operations. We must repeat this recursively $n-1$ times, so the total computational complexity of computing $p(x_1|\boldsymbol{y})$ is $O(nN^2)$. In contrast, the direct (and naive) $n-1$ dimensional integration would require $O(N^{n-1})$ operations.

To illustrate the idea a bit more explicitly, consider the case where $n = 3$.

$$p(\boldsymbol{x}|\boldsymbol{y}) \;\propto\; p(y_1|x_1)\, p(x_1)\, p(y_2|x_2)\, p(x_2|x_1)\, p(y_3|x_3)\, p(x_3|x_2) \;.$$

Let us first compute the posterior distribution of x_1 by "integrating out" x_2 and x_3 (called *marginalization*):

$$\begin{aligned}
p(x_1|\boldsymbol{y}) \quad &\propto \quad \int_{x_2,x_3} p(y_1|x_1)\, p(x_1)\, p(y_2|x_2)\, p(x_2|x_1)\, p(y_3|x_2)\, p(x_3|x_3)\, dx_2 dx_3 \\
&= \quad p(y_1|x_1)\, p(x_1) \int_{x_2} p(y_2|x_2)\, p(x_2|x_1) \left(\int_{x_3} p(y_3|x_3)\, p(x_3|x_2)\, dx_3 \right) dx_2 \\
&= \quad p(y_1|x_1)\, p(x_1) \int_{x_2} p(y_2|x_2)\, p(x_2|x_1)\, q(x_2)\, dx_2 \\
&= \quad p(y_1|x_1)\, p(x_1)\, q(x_1)
\end{aligned}$$

More generally, the process begins by marginalizing over all variables except the root x_1 (i.e., compute the marginal posterior distribution of x_1).

$$p(x_1|\boldsymbol{y}) \quad \propto \quad \int_{x_2,\ldots,x_n} \prod_{i=1}^{n} p(y_i|x_i)\, p(x_i|x_{\rho(i)})\, dx_2 \cdots dx_n \;=\; p(y_1|x_1)\, q(x_1) \;,$$

where

$$q(x_1) \quad = \quad p(x_1) \int_{x_2,\ldots,x_n} \prod_{i=2}^{n} p(y_i|x_i)\, p(x_i|x_{\rho(i)})\, dx_2 \cdots dx_n \;.$$

Because of the factorize form of the integrand, this multidimensional integration can be computed recur-

sively, first integrating with respect to x_n, then x_{n-2} and so on, analogous to the case of a Markov chain above.

2.2 Cyclic Graphs and Random Fields

The distributions of cyclic graphs and random fields do not have a simple factorization. Consequently, efficient algorithms for exactly computing posterior distributionsare not possible. There are a variety of methods to approximate the posterior distribution. Message passing or belief propagation algorithms can be applied iteratively, for example by cycling through the variables repeatedly until the posterior distributions approximately converge. There are no general guarantees that this process will converge to the correct posterior distributions, but this so-called "loopy belief propagation" often produces good results in practice. Another common approach to approximate the posterior distribution is through Monte Carlo procedures such as the Metropolis-Hastings algorithm or Gibbs sampling. The basic idea in these methods is to generate random samples using a cleverly designed Markov chain that is easy to generate samples from and has a stationary distribution that is equal to the desired posterior. In this way, the samples generated from the Markov chain are eventually (approximately) equivalent to samples from the posterior. These samples can then be used to estimate the posterior and/or its moments.

3 Parsimonious Graphical Models

As discussed in the introduction, graphical models can provide relatively simple or low-dimensional models. Roughly speaking, the complexity of a graphical model is related to the number of edges. This notion of complexiy becomes completely transparent in the case of Gaussian graphical models.

First consider a Gaussian Markov chain model with $x_1 \sim \mathcal{N}(0, \sigma_1)$

$$x_i = \theta_i x_{i-1} + \sigma_i \epsilon_i \ , \ i = 2, \ldots, n$$

where $\theta_i \in (0, 1)$ and $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1)$, $i = 1, \ldots, n$. Each $\theta_i \in \mathbb{R}$ is a parameter of the model. This graph of this model is exactly the one shown in Fig. 2. Each edge is associated with one parameter, and represents the conditional probability

$$p_i(x_{i+1}|x_i) = \frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}(x_{i+1} - \theta_i x_i)^2/\sigma_i^2}$$

If the θ_i and σ_i are free parameters, then the model has $2n$ parameters. However, it is common to assume that the Markov chain dependence is the same for each i, so that $\theta_i = \theta$ and $\sigma_i = \sigma$, a single global constant. In this case the model has only two parameters.

Note that $\mathbb{E}[x_i] = 0$ and in the case where the $\theta_i = \theta$ are all equal the covariance of x_i and x_j is $\mathbb{E}[x_i x_{i+j}] = \mathbb{E}[x_i(\theta x_{i+j-1} + \epsilon_j)] = \mathbb{E}[x_i(\theta(\theta x_{i+j-2} + \epsilon_{j-1}) + \epsilon_j)] = \theta^j \mathbb{E}[x_i^2]$. Also, if we assume $\sigma^2 = 1 - \theta^2$, then $\mathbb{E}[x_i^2] = \theta^2 \mathbb{E}[x_{i-1}^2] + (1 - \theta^2)\mathbb{E}[\epsilon_i^2]$. So if $\mathbb{E}[x_1^2] = 1$ and $\mathbb{E}[\epsilon_i^2] = 1$, $i > 1$, we have

$\mathbb{E}[x_i^2] = 1$ for $i > 1$. In this case, the covariance matrix has a simple structure:

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & \theta & \theta^2 & \dots & \theta^{n-1} \\ \theta & 1 & \theta & \cdots & \theta^{n-2} \\ \theta^2 & \theta & 1 & \cdots & \theta^{n-3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta^{n-1} & \theta^{n-2} & \theta^{n-3} & \dots & 1 \end{bmatrix}$$

and $x \sim \mathcal{N}(0, \mathbf{\Sigma})$. The structure of the graph is revealed by considering the inverse covariance matrix, which has a triadiagonal form (zero everywhere except on the diagonal and first off-diagonals). This is simply a consequence of the fact that each variable is conditionally independent given its neighbors in the graph.

An easy way to see this structure is to re-write the graphical model in vector form as follows. Note that

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \theta & 0 & 0 & \cdots & 0 \\ 0 & \theta & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Re-arranging terms, we have $(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x} = \boldsymbol{\epsilon}$ and $\boldsymbol{x} = (\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{\epsilon}$. Thus, we see that $\boldsymbol{x} \sim \mathcal{N}(0, \sigma^2(\boldsymbol{I} - \boldsymbol{A})^{-I}(\boldsymbol{I} - \boldsymbol{A}^T)^{-1})$. This shows that the covariance of x is given by $\mathbf{\Sigma} := \sigma^2(\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A}^T)^{-1}$. The inverse covariance is

$$\begin{aligned} \mathbf{\Sigma}^{-1} &= \frac{1}{\sigma^2}(\boldsymbol{I} - \boldsymbol{A}^T)(\boldsymbol{I} - \boldsymbol{A}) \\ &= \frac{1}{\sigma^2}(I - \boldsymbol{A} - \boldsymbol{A}^T + \boldsymbol{A}^T\boldsymbol{A}) \end{aligned}$$

Note that $\boldsymbol{A}$ is zero except on the first (lower) off-diagonal, $\boldsymbol{A}^T$ is zero except on the first (upper) off-diagonal, and $\boldsymbol{A}^T\boldsymbol{A}$ is diagonal. Therefore, the matrix $\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{A}^T + \boldsymbol{A}^T\boldsymbol{A}$ is tridiagonal. This is simply a consequence of the fact that each variable is conditionally independent given its neighbors in the graph.

Note that the inverse covariance takes the form $\mathbf{\Sigma}^{-1} = \frac{1}{\sigma^2}(\boldsymbol{I} - \boldsymbol{A} - \boldsymbol{A}^T + \boldsymbol{A}^T\boldsymbol{A})$ for any model of the form $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$, with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$. The matrix $\boldsymbol{A}$ defines the structure of the graph and dictates the locations of non-zeros in the inverse covariance. The structure of the graph is revealed by the sparsity pattern of the inverse covariance.

Let's contrast this graphical model with a subspace model. A Gaussian subspace model would assume that $\boldsymbol{x} = \boldsymbol{H}\boldsymbol{\theta}$, where $\boldsymbol{H} \in \mathbb{R}^{n \times k}$ and $\boldsymbol{\theta} \sim \mathcal{N}(0, \boldsymbol{I})$. In this case, the covariance is $\mathbf{\Sigma} = \boldsymbol{H}\boldsymbol{H}^T$ and it has rank k, and thus this model requires nk parameters (which define the subspace). The rank of the covariance of the graphical model above is n, but it is defined by only 2 parameters, θ and σ. So if our signal has a Markovian structure, the subspace model is inappropriate. It would require $O(n^2)$ parameters to model the same covariance.

4 Learning the Graph Structure

Many methods have been proposed for estimating graphical models from data. Here we will focus on learning Gaussian graphical models. There are extensions of the techniques discussed below to non-Gaussian models, but we will not discuss them here. Recall the following Gaussian Markov model:

$$\boldsymbol{x} \;=\; \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Above, we showed that $\boldsymbol{\Sigma} := \sigma^2(\boldsymbol{I} - \boldsymbol{A})^{-1}(\boldsymbol{I} - \boldsymbol{A}^T)^{-1}$ and therefore the inverse covariance is

$$\begin{aligned} \boldsymbol{\Sigma}^{-1} &= \frac{1}{\sigma^2}(\boldsymbol{I} - \boldsymbol{A}^T)(\boldsymbol{I} - \boldsymbol{A}) \\ &= \frac{1}{\sigma^2}(\boldsymbol{I} - \boldsymbol{A}^T - \boldsymbol{A} + \boldsymbol{A}^T\boldsymbol{A}) \end{aligned}$$

The sparsity pattern in the matrix $\boldsymbol{A}$ defines the structure of the graph and dictates the locations of non-zeros in the inverse covariance. The structure of the graph is revealed by the sparsity pattern of the inverse covariance.

More generally, the edge pattern of any Gaussian graphical model is indicated by the non-zero entries the inverse of its covariance matrix. Let $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ be a Gaussian graphical model. The nonzero entries in the inverse covariance $\boldsymbol{\Sigma}^{-1}$ determine the edges in the graph. Obviosly, one way to learn the graphical model is to estimate the covariance and/or its inverse. An accurate estimate will require a large number of i.i.d. samples $\{\boldsymbol{x}_k\}_{k=1}^m \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. Since the covariance is $n \times n$, a simple accounting suggests that in order to estimate the $O(n^2)$ covariances will require at least $O(n^2)$ measurments. Each observation $\boldsymbol{x}_k \in \mathbb{R}^n$ provides n scalar values, and so $m = O(n)$ such observations is a bare minimum requirement to estimate a general $n \times n$ covariance.

But what if we believe that the graphical model is *sparse* in the sense that there are far fewer than $O(n^2)$ edges in the graph. That is, suppose each variable/vertex is connected to at most $d \ll n$ other variables/vertices. Then there are only $O(dn)$ edges in the graph. Since the graphical model is determined by the weights on these edges, in such a case we might be able to get by far fewer measurements than needed for the case of a generic n-dimensional covariance.

It turns out that the structure of the graph can be estimated more efficiently by solving a series of regression problems. To see this, suppose we are interested in predicting one of the variables in $\boldsymbol{x}$, say x_i, based on all the other variables. Since the $x_1, \ldots, x_n$ are jointly Gaussian, we know that the optimal predictor will be a linear function of the form $\widehat{x}_i = \sum_{j \neq i} \theta_{ij} x_j$, for some weights $\{\theta_{ij}\}$.

Lemma 1. *The optimal weights are given by*

$$\theta_{ij} = -\boldsymbol{\Sigma}^{-1}_{ij} / \boldsymbol{\Sigma}^{-1}_{ii} \; . \tag{1}$$

This is a nice formula, since it shows very clearly that $\theta_{ij} = 0$, whenever $\boldsymbol{\Sigma}^{-1}_{ij} = 0$. If each variable/vertex is connected to at most k other vertices, then this implies that the coefficient set $\{\theta_{ij}\}$ will be d-sparse. Therefore, we learn the structure of the graph by performing n sparse regressions (e.g., using the lasso).

Proof. The formula (1) can be obtained by the method of Lagrange multipliers as follows. Let $\boldsymbol{e}_i$ denote the canonical unit vector with 1 in the ith location and 0 elsewhere. The problem of predicting x_i using the

other variables can be written as

$$\min_{\boldsymbol{\theta}} \mathbb{E}[(\boldsymbol{x}^T \boldsymbol{e}_i - \boldsymbol{x}^T \boldsymbol{\theta})^2] \text{ , subject to } \boldsymbol{e}_i^T \boldsymbol{\theta} = 0 \text{ .}$$

Note that the function we are trying to minimize is

$$\mathbb{E}[(\boldsymbol{x}^T \boldsymbol{e}_i - \boldsymbol{x}^T \boldsymbol{\theta})^2] \;=\; \boldsymbol{e}_i^T \boldsymbol{\Sigma} \boldsymbol{e}_i - 2 \boldsymbol{e}^T \boldsymbol{\Sigma} \boldsymbol{\theta} + \boldsymbol{\theta}^T \boldsymbol{\Sigma} \boldsymbol{\theta} \text{ .}$$

We can ignore the first term since it is independent of $\boldsymbol{\theta}$. Thus, the Lagrangian for this constrained optimization is

$$L(w, m) \;=\; -2 \boldsymbol{e}_i^T \boldsymbol{\Sigma} \boldsymbol{\theta} + \boldsymbol{\theta}^T \boldsymbol{\Sigma} \boldsymbol{\theta} + \lambda \boldsymbol{e}_i^T \boldsymbol{\theta},$$

where λ is the Lagrange multiplier. The partial derivative of L with respect to $\boldsymbol{\theta}$ is

$$\frac{\partial L}{\partial \boldsymbol{\theta}} \;=\; -2 \boldsymbol{\Sigma} \boldsymbol{e}_i + 2 \boldsymbol{\Sigma} \boldsymbol{\theta} + \lambda \boldsymbol{e}_i \text{ .}$$

Setting this to zero we have

$$\boldsymbol{\theta} \;=\; \boldsymbol{\Sigma}^{-1} (\boldsymbol{\Sigma} \boldsymbol{e}_i - \lambda \boldsymbol{e}_i / 2) \;=\; \boldsymbol{e}_i - \lambda \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i / 2 \text{ .}$$

The partial derivative of L with respect to λ is

$$\frac{\partial L}{\partial \lambda} \;=\; \boldsymbol{e}_i^T \boldsymbol{\theta} \text{ .}$$

Setting this to zero gives us the constraint condition $\boldsymbol{e}_i^T \boldsymbol{\theta} = 0$. Now apply this condition to the solution of $\boldsymbol{\theta}$ above

$$0 \;=\; \boldsymbol{e}_i^T \boldsymbol{\theta} \;=\; \boldsymbol{e}_i^T (\boldsymbol{e}_i - \lambda \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i / 2) = 1 - \lambda \boldsymbol{e}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i / 2 \text{ .}$$

This give us the solution for λ:

$$\lambda \;=\; 2 / (\boldsymbol{e}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i) \text{ .}$$

Plugging this back into the expression for $\boldsymbol{\theta}$ we get the solution:

$$\boldsymbol{\theta} \;=\; \boldsymbol{e}_i - \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i / (\boldsymbol{e}_i^T \boldsymbol{\Sigma}^{-1} \boldsymbol{e}_i) \text{ .}$$

□

We have shown that if the degree of each vertex is at most d, then the optimal regression weights will be d-sparse. This suggests the following approach to learning the structure of the graph. Let $x_{i,t}$ denote the sample of the ith variable in the tth observation vector $\boldsymbol{x}_t$, $t = 1, \ldots, m$.

for $i = 1, \ldots, n$

let $\boldsymbol{\theta}_i$ **be the solution to**

$$\min_{\boldsymbol{\theta} \,:\, \boldsymbol{e}_i^T \boldsymbol{\theta} = 0} \sum_{t=1}^{m} (x_i - \boldsymbol{\theta}^T \boldsymbol{x}_t)^2 \;+\; \lambda \|\boldsymbol{\theta}\|_1$$

estimate the edge set for vertex i **to be** $\{j \;:\; j \in \textbf{support}(\boldsymbol{\theta}_i)\}$

end

Recall that under suitable conditions, the lasso can estimate the correct d-sparse weight vector very accurately with $O(d \log n)$ samples. This shows that the algorithm proposed above may be able to recover the correct graph structure with $m = O(d \log n)$, where m is the number of i.i.d. samples from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. In contrast, the naive approach that does not exploit sparsity requires $O(n^2)$ samples.