Gaussian Graphical Models

When the RVs in the nodes of the graph are the marginals of a Gaussian with covariance matrix Σ, then the entries of Σ^{-1}, which are 0, correspond exactly to the edges **not** contained in the independence graph. We call this a Gaussian Graphical Model.
In fact, the missing edges denote exactly the conditional independencies.
So it is “easy” to compute the independence graph, because the pairwise Markov property is fulfilled.
We’ll say more about “easy” later, because this matrix inversion is not always so easy.

Partial correlation coefficient

The partial correlation coefficient between X_1 and X_2 is the correlation coefficient of the residual vectors arising from regression of X_1 vs the $X_3, \ldots, X_K$, and the residual vectors arising from regressing X_2 vs $X_3, \ldots, X_k$.

Partial correlation coefficients can be computed from the entries of the inverse of the variance-covariance matrix: Theorem: Let $X_1, \ldots., X_k$ be random vectors and $P = \Sigma^{-1}$.

$$\rho_{i,j|rest} = -\frac{p_{ij}}{\sqrt{p_{ii}p_{jj}}}$$

Singular value decomposition

Covariance matrix is the product of (normalized) data matrix times its transpose.
Depending on the shape (rank) of the data matrix, the product will be singular. How to invert a singular matrix?
Singular value decomposition of matrix A , of shape $n \times m$:

$$A = UWV^T$$

U is $m \times m$ and contains the eigenvectors of AA^T.
W is a diagonal matrix and contains the singular values. They are the square roots of the eigenvalues of AA^T (which are also the eigenvalues of $A^T A$).
V is $n \times n$.
Condition number: defined as the largest singular value divided by the smallest one. If large, we say the matrix is ill-conditioned.

Pseudoinverse

$$A^+ = VW^{-1}U^T$$

We are applying this specifically to a quadratic matrix (the covariance matrix).In this case the W^{-1} is contains the diagonal values $1/w_i$. They trick is that the singular values on the diagonal of W are sorted and the last ones are 0 (singular matrix) or near-zero (ill-conditioned matrix). The inversion is done by only taking the inverse for the the singular values that are not 0, or are sufficiently different from 0.
See SVD function in R, or in package corpcor (Strimmer).

Shrinkage methods: Strimmers method

See R packages corpcor and GeneNet. Estimate a b̈etterc̈ovariance matrix. Mix the covariance matrix with another matrix (e.g., for a multivariate Gaussian, with little weight off-diagonal).
Let U be the observed covariance matrix, and T be a (full rank) covariance matrix that might look like it could have generated the data.

$$\hat{U} = \lambda T + (1 - \lambda) U$$

$\hat{U}$ will be invertible and not change reality too much. See Schäfer and Strimmer, 2005.

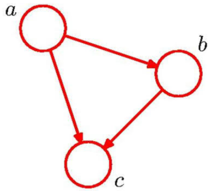
Target A: "diagonal, unit variance" 0 estimated parameters $t_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij}) + \sum_i \widehat{\mathrm{Var}}(s_{ii})}{\sum_{i\neq j} s_{ij}^2 + \sum_i (s_{ii}-1)^2}$	**Target B:** "diagonal, common variance" 1 estimated parameter: v $t_{ij} = \begin{cases} v = \mathrm{avg}(s_{ii}) & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij}) + \sum_i \widehat{\mathrm{Var}}(s_{ii})}{\sum_{i\neq j} s_{ij}^2 + \sum_i (s_{ii}-v)^2}$
Target C: "common (co)variance" 2 estimated parameters: v, c $t_{ij} = \begin{cases} v = \mathrm{avg}(s_{ii}) & \text{if } i = j \\ c = \mathrm{avg}(s_{ij}) & \text{if } i \neq j \end{cases}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij}) + \sum_i \widehat{\mathrm{Var}}(s_{ii})}{\sum_{i\neq j} (s_{ij}-c)^2 + \sum_i (s_{ii}-v)^2}$	**Target D:** "diagonal, unequal variance" p estimated parameters: s_{ii} $t_{ij} = \begin{cases} s_{ii} & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij})}{\sum_{i\neq j} s_{ij}^2}$
Target E: "perfect positive correlation" p estimated parameters: s_{ii} $t_{ij} = \begin{cases} s_{ii} & \text{if } i = j \\ \sqrt{s_{ii}s_{jj}} & \text{if } i \neq j \end{cases}$ $f_{ij} = \frac{1}{2}\{ \sqrt{\frac{s_{jj}}{s_{ii}}} \widehat{\mathrm{Cov}}(s_{ii}, s_{ij}) + \sqrt{\frac{s_{ii}}{s_{jj}}} \widehat{\mathrm{Cov}}(s_{jj}, s_{ij}) \}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij}) - f_{ij}}{\sum_{i\neq j} (s_{ij} - \sqrt{s_{ii}s_{jj}})^2}$	**Target F:** "constant correlation" $p + 1$ estimated parameters: s_{ii}, $\bar{r}$ $t_{ij} = \begin{cases} s_{ii} & \text{if } i = j \\ \bar{r}\sqrt{s_{ii}s_{jj}} & \text{if } i \neq j \end{cases}$ $\hat{\lambda}^\star = \frac{\sum_{i\neq j} \widehat{\mathrm{Var}}(s_{ij}) - \bar{r} f_{ij}}{\sum_{i\neq j} (s_{ij} - \bar{r}\sqrt{s_{ii}s_{jj}})^2}$

Bayesian Networks

Given a directed acyclic graph (DAG) G.
Associate to each node a distribution, that is defined as a conditional distribution on the distributions in the parent nodes.
("Parent node" is a well-defined notion due to the DAG structure.)
The joint distribution can be computed from the conditional probability distributions in the nodes ("local probability models").
Then, this model (DAG plus local probability models) is called a Bayesian Network.
DAG is used to represent the factorization of a distribution. Think of the conditional probability distributions as (little) tables. When a node has several parents, this will be multi-dimensional table.

Bayesian Networks

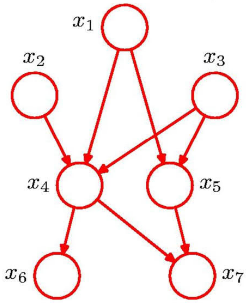
Directed Acyclic Graph (DAG)



$$p(a, b, c) = p(c|a, b)p(a, b) = p(c|a, b)p(b|a)p(a)$$

$$p(x_1, \ldots, x_K) = p(x_K|x_1, \ldots, x_{K-1}) \ldots p(x_2|x_1)p(x_1)$$

Bayesian Networks



$$\begin{aligned} p(x_1,\ldots,x_7) &= p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3) \\ &\quad p(x_5|x_1,x_3)p(x_6|x_4)p(x_7|x_4,x_5) \end{aligned}$$

General Factorization

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathrm{pa}_k)$$

Conditional independence in a BN G:
A variable X is conditionally independent of its "non-descendants" given the parents of X. Compare to the local Markov condition. This is a directed version.

And, loosely formulated:
When X and Y are conditionally independent with respect to Z, then the BN G will not contain an edge between X and Y.

Factorization

Let G be a BN over the variables $X_1, ..., X_n$. We say that a (high-dimensional) distribution P factorizes according to G if P can be expressed as the product of the conditional probability distributions:

$$P(X_1, ..., X_n) = \prod P_{local}(X_i | parents(X_i))$$

Goal

Given a high-dimensional distribution. Find a BN (DAG plus local distributions) such that its joint distribution **approximates** the given high-dimensional distribution.
BN is a generative model.
This is also efficient: For 2-state variables in each cell, the joint distribution is defined by 2^k entries. If we can represent it as a BN, we save on parameters (compute!).

Equivalence classes

Some graph motifs are indistinguishable when it comes to the distribution they generate. Leads to equivalence classes. (Board!)

Theorem 2.1 (Pearl and Verma, 1991). Two DAGs are equivalent if and only if they have the same underlying undirected graph and the same v-structures (i.e., converging directed edges into the same node, such that $a \to b \leftarrow c$, and there is no edge between a and c).
Moreover, an equivalence class of network structures can be uniquely represented by a partially directed graph (PDAG), where a directed edge $X \to Y$ denotes that all members of the equivalence class contain the arc X$\to Y$; an undirected edge $X - Y$ denotes that some members of the class contain the arc $X \to Y$, while others contain the arc $Y \leftarrow X$. (cited from Friedman et al., JCB 2000)

Tasks

- Inference:
- Learning: Estimate parameters of LPD!
- Structure learning: What is a good network?

Inference

Am I out of fuel?
Relies on Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

For an example, see slides by Bishop ("Am I out of fuel?")
`https://www.microsoft.com/en-us/research/wp-content/uploads/2016/05/prml-slides-8.pdf`

For later, general formula for Bayes theorem: Let A be partitioned into subsets A_i. Then $P(B) = \sum_j P(B|A_j$, and therefore:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_j P(B|A_j}$$

Learning in BNs

Given a high-dimensional distribution and a DAG G. We assume a particular distribution for the LPDs, depending on a parameter (vector). We want to infer the parameter.
Concrete example: Let all variables be binary. LDP: binomial distribution, little tables. Parameter θ_i for each binomial. We want to infer θ.

Bayesian thinking

D - Daten, G - DAG Prior distribution on θ: $P(\theta)$ $(= P(\theta|G))$.

$$P(\theta|D, G) = \frac{P(D|\theta, G)P(\theta|G)}{P(D|G)}$$

$P(D|\theta, G)\ldots$ data likelihood
$P(D|G)\ldots$ model evidence
$P(\theta|D, G)\ldots$ posterior (distribution of θ)

Parameter estimation

If we wanted to estimate θ in the Bayesian setting we would use the expectation over the posterior distribution of θ.
This is in contrast to maximum likelihood estimation, where the argmax is used.

Probability of the data

For a reasonable estimate of the probability of the data under the model we compute its expectation with respect to θ (=marginal likelihood, integrating out the parameter θ). This is the denominator from above.

$$P(D|G) = \int P(D,\theta|G)d\theta = \int P(D|\theta,G)P(\theta|G)d\theta$$

G remains fixed.

Alternative view

$P(G|D) \ldots$ Bayes score (of a graph)

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}$$

For the purpose of comparing different graphs, $P(D)$ doesn't matter, because it is the same for all graphs. This is convenient, since we anyway do not know how to compute it (it would include averaging over all models).
$P(G)$ we assume uniform, i.e. all graphs are equally likely to occur. Then, in order to compare across graph structures, we find that $P(G|D)$ is proportional to $P(D|G)$ - and this we know how to compute.

Summary

Model evidence:
prior $\rightarrow$ model evidence

Parameter estimation:
prior $\rightarrow$ model evidence $\rightarrow$ posterior $\rightarrow$ expectation $\hat{\theta}$

Model evidence is an alternative to the maximal likelihood of the data. While the maximum of the likelihood is assumed at the maximizing θ, there s no particular θ associated to the model evidence.

Choice of prior

Assume a BN with a 2 possible outcomes in each node (variable). E.g., a gene is expressed HIGH or LOW.
Then each conditional distribution can be modeled as a binomial distribution with parameter θ_i, indexed over all edges (2 per edge, I think).

The beta distribution

Beta distribution is the continuous analogue to the binomial. The density function f of a beta distribution $\beta(\alpha, \beta)$ is

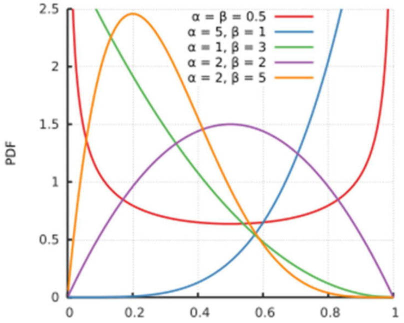
$$f(x; \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

Remember the gamma function. It is the continuous extension of the factorial:

$$\Gamma(n) = (n-1)!$$

We might need:

$$\int_0^1 (1-x)^{\alpha-1} x^{\beta-1} dx = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$


PDF
α = β = 0.5
α = 5, β = 1
α = 1, β = 3
α = 2, β = 2
α = 2, β = 5
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1

Binomial setting. H heads, T tails observed ($= D$). Let θ be distributed according to a Beta distribution $\beta(\theta; \alpha, \beta)$. Then one obtains the following posterior (board):

$$P(\theta|D) = \beta(\theta; \alpha + H, \beta + T)$$

So assuming a beta prior the posterior is again beta. We say the "beta is the conjugate prior to the binomial distribution".
For the estimate $\hat{\theta}$ one obtains (board):

$$\hat{\theta} = \int \theta P(\theta|D) d\theta = \frac{\alpha + H}{\alpha + \beta + H + T}$$

Dirichlet distribution is the conjugate prior to the multinomial

Think of a multinomial distribution with k classes and probabilities $\theta = (p_1, \dots, p_k)$.
Dirichlet distribution:

$$\mathcal{D}(\theta; \alpha_1, \dots, \alpha_k) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \prod \theta_i^{\alpha_i - 1}$$

And with this prior

$$P(\theta|D) = \mathcal{D}(\theta; \alpha_1 + N_1, \dots, \alpha_k + N_k)$$

Computing the model evidence for a BN: factorization

Keep G fixed. To compute the model evidence, we need the integral

$$P(D|G) = \int P(D|\theta, G)P(\theta|G)d\theta$$

In a BN this factorizes over the LPDs! (Assume independence of parameters.)

Therefore we can compute the model evidence for any graph G.

In (2.99) we use the fact of parameter independence from (2.96). The last step is performed analogously to (3.61).

Structural Learning

In the beginning of this section we have already stated that we want to retrieve the independency structure of the random variables. In (2.79) we have defined the estimator of the best structure as $\lambda^* = \operatorname{argmax}_\lambda P(\lambda|\mathbf{x})$. Therefore, we have to iterate over all $\lambda \in \{\lambda_1, \ldots, \lambda_m\}$ to compute the posterior

$$\begin{aligned} P(\lambda|\mathbf{x}) &= \frac{P(\mathbf{x}|\lambda) \cdot P(\lambda)}{P(\mathbf{x})} \\ &\propto P(\mathbf{x}|\lambda) \cdot P(\lambda) \end{aligned} \tag{2.100}$$

Since $P(\mathbf{x})$ is independent of the chosen structure λ we can use the proportionality. Without prior knowledge of the structure, we can use the uniform distribution. Therefore, we can neglect $P(\lambda)$. Using the Dirichlet distribution as parameter prior, we can analytically solve $P(\mathbf{x}|\lambda)$ in (2.100):

$$\begin{aligned} P(\mathbf{x}|\lambda) &= \int P(\mathbf{x}|\theta,\lambda) \cdot P(\theta|\lambda)\, \mathrm{d}\theta \\ &= \int \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\left(\sum_{\iota=1}^{r} N_\iota\right)!}{\prod_{\iota=1}^{r} N_\iota!} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \cdot \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}\, \mathrm{d}\theta \\ &= \frac{\left(\sum_{\iota=1}^{r} N_\iota\right)!}{\prod_{\iota=1}^{r} N_\iota!} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \int \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}+\alpha_{ijk}-1}\, \mathrm{d}\theta_{ij} \\ &= \frac{\left(\sum_{\iota=1}^{r} N_\iota\right)!}{\prod_{\iota=1}^{r} N_\iota!} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \cdot \frac{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk} + N_{ijk}\right)}{\Gamma\left(\sum_{k=1}^{r_i} \left(\alpha_{ijk} + N_{ijk}\right)\right)} \\ &= \frac{\left(\sum_{\iota=1}^{r} N_\iota\right)!}{\prod_{\iota=1}^{r} N_\iota!} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\Gamma\left(\sum_{k=1}^{r_i} \left(\alpha_{ijk} + N_{ijk}\right)\right)} \cdot \prod_{k=1}^{r_i} \frac{\Gamma\left(\alpha_{ijk} + N_{ijk}\right)}{\Gamma\left(\alpha_{ijk}\right)} \end{aligned} \tag{2.101}$$

After expanding with θ, we reduce the global problem to local ones. We compute the probability for each node ($i = 1, \ldots, n$), for each configuration of the parent nodes ($j = 1, \ldots, q_i$), and for each state of the actual node ($k = 1, \ldots, r_i$) individually. Next, we shift some terms out of the integral. Using (3.59) in the appendix 3.4.2 on page 46, we can solve the integral. Finally, we reorganize the equation.

Applying (2.101) to each $\lambda \in \{\lambda_1, \ldots, \lambda_m\}$, we can estimate the best structure. As $\{\lambda_1, \ldots, \lambda_m\}$ contains all DAGs with the same number of nodes as the network, this set is huge and make the computations intractable. After clarifying this statement, we present MCMC as approximation

Comparing across graphs: BIC

It is difficult to compare models with different numbers of parameters because the model with more parameters (edges) will always give a "better" explanation of the data, i.e. having higher probability. Remedy: Penalize the likelihood.

Let d be the number of parameters and N the number of data points (the sample size)
Bayesian information criterion (BIC):

$$score_{BIC}(G) = \max_{\theta} \left(P(D|\theta, G) - \frac{d}{2} \log(N) \right)$$

The "better" model is the one with the higher BIC-score.

Comparing across graphs

For each graph, we can in principle compute its Bayes score. But there are MANY DAGs on n nodes $\rightarrow$ stochastic search strategy: Markov Chain Monte Carlo (MCMC)

MCMC comes in different flavors: Gibbs sampling, Metropolis-Hastings, etc.

MCMC: Metropolis-Hastings algorithm

Ingredient: A procedure "modify", which adds/swaps/deletes edges, and secures that the resulting graph is again cycle-free (so as to remain in the realm of DAGs).

Start: Take a DAG G_0 and calculate its probability $P(G_0|D)$.
$t \leftarrow 0$

While (there is still coffee in the mug):
$G_{t'} \leftarrow modify(G_t)$
Accept new DAG with probability

$$\min(1, \frac{P(G_{t'}|D)}{P(G_t|D)}).$$

If accepted $G_{t+1} \leftarrow G_{t'}$ and $t \leftarrow t+1$.
end while.
Output G_t.

Theorem: This algorithm converges (after a lot of coffee) to the optimal G.

Why?: With the suggested acceptance probability, the accept/reject decision defines a Markov chain on the space of DAGs. By applying steps of this Markov chain one ensure convergence to the MC's equilibrium distribution. This means that after a burn-in phase, the algorithm explores this distribution, and thus samples the more likely DAGs more often.

Remark: In reality, MCMC is better for integration than for optimization.

It is better in the end not to select one DAG, but to look for frequently used edges.

Problem: A graph that is made up of edges occurring frequently in the MCMC run need not be acyclic.

Heuristics to compute BNs: PC algorithm

See "Causation, Prediction, and Search" by Spirtes, Glymour, and Scheines, p116ff. Roughly it works as follows:

- Iterative algorithm
- Starts with complete, undirected graph
- Find edges which are conditionally independent with respect to 1 other variable. Delete.
- Find edges which are CI wrt to 2-element-sets of variables. Delete.
- etc.
- Find and orient the colliders

Issues

- Why do we delete these edges? Because if a BN would contain such a CI edge, this would contradict the directed local Markov condition (A variable X_i is conditionally independent of its "non-descendants" given the parents of X_i.)
- How to test for conditional independence? Partial correlation!
- Does it depend on the order of processing? No and yes.
- How does one orient the edges?

by any subset of variables containing Y but not X, Z, the algorithm will mistakenly require a collision at Y, and this requirement will ramify through orientations of other edges. Or, if the true structure contains a collision at Y but X - Y is omitted in the input to step C), no unique orientation will be given to Y - Z, and this uncertainty may ramify through the orientations of other edges on paths including Z.

Instabilities may also arise in Step C) because of errors in the list of d-separation relations input, even when the underlying undirected graph is correct. If in the input to C), X is adjacent to Y and Y to Z but not X to Z and a d-separation relation between X and Z given $\mathbf{S}$ containing Y is omitted from the input, no orientation error will result unless no other set containing Y d-separates X and Z. But if in the true directed graph, the edges between X and Y and between Y and Z collide at Y, and a d-separation relation involving X and Z and some set $\mathbf{U}$ containing Y but not X or Z is erroneously included in the input, the algorithm will conclude that there is no collision at Y, and this error may be ramified to other edges.

A little reflection on Step C) reveals that its output may not be a collection of directed acyclic graphs if one of the four assumptions listed at the beginning of this section is violated. This is not necessarily a defect of the algorithm. If the algorithm finds that the edges X - Y - Z collide at Y, and Y - Z - W collide at Z, it will create a pattern with an edge Y <-> Z. Double headed edges can occur when the causal structure is not causally sufficient, or when there is an error in input (as from sampling variation). They have a theoretical role in identifying the presence of unmeasured common causes, an issue discussed further in the next chapter.

5.4.2 The PC Algorithm

In the worst case, the SGS algorithm requires a number of d-separation tests that increases exponentially with the number of vertices, as must any algorithm based on conditional independence relations or vanishing partial correlations. But the SGS algorithm is very inefficient because for edges in the true graph the worst case is also the expected case. For any undirected edge that is in the graph G, the number of d-separation tests that must be conducted in stage B) of the algorithm is unaffected by the connectivity of the true graph, and therefore even for sparse graphs the algorithm rapidly becomes infeasible as the number of vertices increases. Besides problems of computational feasibility, the algorithm has problems of reliability when applied to sample data. The determination of higher order conditional independence relations from sample distributions is generally less reliable than is the determination of lower order independence relations. With, say, 37 variables taking three values each, to determine the conditional independence of two variables on the set of all remaining variables requires considering the relations among the frequencies of 3^{35} distinct states, only a fraction of which will be instantiated even in very large samples.

We should like an algorithm that has the same input/output relations as the SGS procedure for faithful distributions but which for sparse graphs does not require the testing of higher order independence relations in the discrete case, and in any case requires testing as few d-separation relations as possible. The following procedure (Spirtes, Glymour, and Scheines, 1991) starts by forming the complete undirected graph, then "thins" that graph by removing edges with zero order conditional independence relations, thins again with first order conditional independence relations, and so on. The set of variables conditioned on need only be a subset of the set of variables adjacent to one or the other of the variables conditioned.

Let **Adjacencies**(C,A) be the set of vertices adjacent to A in directed acyclic graph C. (In the algorithm, the graph C is continually updated, so **Adjacencies**(C,A) is constantly changing as the algorithm progresses.)

PC Algorithm:

```
A.) Form the complete undirected graph $C$ on the vertex set $\mathbf{V}$.
B.)
    $n = 0$.
    repeat
        repeat
            select an ordered pair of variables $X$ and $Y$ that are adjacent in $C$ such
            that Adjacencies($C,X$)\{$Y$} has cardinality greater than or equal to
            $n$, and a subset $\mathbf{S}$ of Adjacencies($C,X$)\{$Y$} of cardinality $n$, and if
            $X$ and $Y$ are d-separated given $\mathbf{S}$ delete edge $X$ - $Y$ from $C$ and
            record $\mathbf{S}$ in Sepset($X,Y$) and Sepset($Y,X$);
        until all ordered pairs of adjacent variables $X$ and $Y$ such that
        Adjacencies($C,X$)\{$Y$} has cardinality greater than or equal to $n$ and all
```

C.) For each triple of vertices X, Y, Z such that the pair X, Y and the pair Y, Z are each adjacent in C but the pair X, Z are not adjacent in C, orient X - Y - Z as X -> Y <- Z if and only if Y is not in **Sepset**(X,Z).

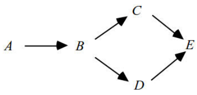
D. repeat

If A -> B, B and C are adjacent, A and C are not adjacent, and there is no arrowhead at B, then orient B - C as B -> C.

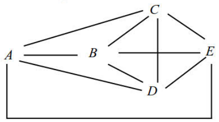
If there is a directed path from A to B, and an edge between A and B, then orient A - B as A -> B.

until no more edges can be oriented.

Figure 1 traces the operation of the first two parts of the PC algorithm:



True Graph



Complete Undirected Graph

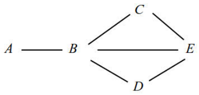
n = 0 No zero order independencies

n = 1 First order independencies

Resulting Adjacencies

$A \perp\!\!\!\perp C \mid B$ $A \perp\!\!\!\perp D \mid B$

$A \perp\!\!\!\perp E \mid B$ $C \perp\!\!\!\perp D \mid B$



n = 2: Second order independencies

Resulting Adjacencies

Although it does not in this case, stage B) of the algorithm may continue testing for some steps after the set of adjacencies in the true directed graph has been identified. The undirected graph at the bottom of figure 1 is now partially oriented in step C). The triples of variables with only two adjacencies among them are:

A - B - C;	A - B -D;
C - B - D;	B - C - E;
B - D - E;	C - E - D

E is not in **Sepset**(C,D) so C - E and E - D collide at E. None of the other triples form colliders. The final pattern produced by the algorithm is shown in figure 2.

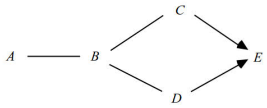


Figure 2

The pattern in figure 2 characterizes a faithful indistinguishability class. Every orientation of the undirected edges in figure 2 is permissible that does not include a collision at B.

5.4.2.1 Complexity

The complexity of the algorithm for a graph G is bounded by the largest degree in G. Let k be the maximal degree of any vertex and let n be the number of vertices. Then in the worst case the number of conditional independence tests required by the algorithm is bounded by

PC algorithm: order of processing

Theorem: If for the given data there exists a BN representation, then the order of processing is unimportant, i.e., any order will recover the correct BN.
Without proof.

For approximation purposes, the order matters.

d-separation

See Bishop slides

Alternative phrasing (from "Causation, prediction, and search"): Following Pearl (1988), we say that for a graph G, if X and Y are vertices in G, $X \neq Y$, and W is a set of vertices in G not containing X or Y, then X and Y are d-separated given W in G if and only if there exists no undirected path U between X and Y, such that (i) every collider on U has a descendent in W and (ii) no other vertex on U is in W.