732A96/TDDE15 Advanced Machine Learning
Graphical Models and Hidden Markov Models

Jose M. Peña
IDA, Linköping University, Sweden

Lecture 4: Structure Learning

Contents

- Structure Learning for BNs
  - Independence Test Based Approach
  - Score Based Approach
- Structure Learning for MNs
  - Independence Test Based Approach

Literature

- Main source
  - Koski, T. J. T. and Noble, J. M. A Review of Bayesian Networks and Structure Learning. *Mathematica Applicanda* 40, 51-103, 2012.

Structure Learning for BNs: Independence Test Based Approach

- We can get a DAG H such that $p(X) = \prod_i p(X_i | pa_H(X_i))$ and, thus, that we can use for probabilistic reasoning as follows:

Let $Y_{1:n}$ be any ordering of the random variables $X_{1:n}$
For each Y_i do
    Set $pa_H(Y_i)$ to be any minimal subset of $Y_{1:i-1}$ such that
    $Y_i \perp_p Y_{1:i-1} \setminus pa_H(Y_i) | pa_H(Y_i)$

Structure Learning for BNs: Independence Test Based Approach

- We can get a DAG H such that $p(X) = \prod_i p(X_i | pa_H(X_i))$ and, thus, that we can use for probabilistic reasoning as follows:

  > Let $Y_{1:n}$ be any ordering of the random variables $X_{1:n}$
  > For each Y_i do
  >   Set $pa_H(Y_i)$ to be any minimal subset of $Y_{1:i-1}$ such that
  > $Y_i \perp_p Y_{1:i-1} \setminus pa_H(Y_i) | pa_H(Y_i)$

- **Exercise**. Prove the previous statement.

Structure Learning for BNs: Independence Test Based Approach

- We can get a DAG H such that $p(X) = \prod_i p(X_i | pa_H(X_i))$ and, thus, that we can use for probabilistic reasoning as follows:

  Let $Y_{1:n}$ be any ordering of the random variables $X_{1:n}$
  For each Y_i do
      Set $pa_H(Y_i)$ to be any minimal subset of $Y_{1:i-1}$ such that
      $Y_i \perp_p Y_{1:i-1} \smallsetminus pa_H(Y_i) | pa_H(Y_i)$

- **Exercise**. Prove the previous statement.
- Note that H has the minimum number of edges among the DAGs that are consistent with the ordering considered.

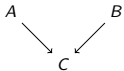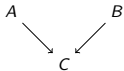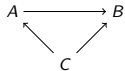## Structure Learning for BNs: Independence Test Based Approach

- We can get a DAG H such that $p(X) = \prod_i p(X_i | pa_H(X_i))$ and, thus, that we can use for probabilistic reasoning as follows:

  > Let $Y_{1:n}$ be any ordering of the random variables $X_{1:n}$
  > For each Y_i do
  >   Set $pa_H(Y_i)$ to be any minimal subset of $Y_{1:i-1}$ such that
  > $Y_i \perp_p Y_{1:i-1} \setminus pa_H(Y_i) | pa_H(Y_i)$

- **Exercise**. Prove the previous statement.
- Note that H has the minimum number of edges among the DAGs that are consistent with the ordering considered.
- However, H may not have the minimum number of edges among all the DAGs, i.e. the ordering considered may not be optimal.

$A \perp_p B$	H with ordering A, B, C	H with ordering C, A, B

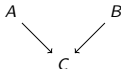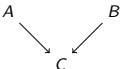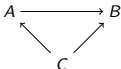## Structure Learning for BNs: Independence Test Based Approach

- We can get a DAG H such that $p(X) = \prod_i p(X_i | pa_H(X_i))$ and, thus, that we can use for probabilistic reasoning as follows:

  > Let $Y_{1:n}$ be any ordering of the random variables $X_{1:n}$
  > For each Y_i do
  >   Set $pa_H(Y_i)$ to be any minimal subset of $Y_{1:i-1}$ such that
  > $Y_i \perp_p Y_{1:i-1} \smallsetminus pa_H(Y_i) | pa_H(Y_i)$

- **Exercise**. Prove the previous statement.
- Note that H has the minimum number of edges among the DAGs that are consistent with the ordering considered.
- However, H may not have the minimum number of edges among all the DAGs, i.e. the ordering considered may not be optimal.

$A \perp_p B$	H with ordering A, B, C	H with ordering C, A, B

- We can get one such optimal DAG H without searching over the $n!$ orderings assuming **faithfulness**, i.e. $U \perp_p V | Z$ if and only if $U \perp_G V | Z$ for some DAG G.

Structure Learning for BNs: Independence Test Based Approach

Parents and children (PC) algorithm

Let H be the complete undirected graph
$l := 0$
Repeat while $l \le n - 2$
    For each ordered pair of nodes X_i and X_j in H such that $X_i \in ad_H(X_j)$ and $|ad_H(X_i) \smallsetminus X_j| \ge l$
        If there is some $S \subseteq ad_H(X_i) \smallsetminus X_j$ such that $|S| = l$ and $X_i \perp_p X_j | S$, then
            $S_{ij} := S_{ji} := S$
            Remove the edge $X_i - X_j$ from H
    $l := l + 1$
Apply the rule R1 to H while possible
Apply the rules R2-R4 to H while possible

R1: $X_i \relbar X_j \relbar X_k$ $\Rightarrow$ $X_i \longrightarrow X_j \longleftarrow X_k$
                  $\wedge \; X_j \notin S_{ik}$

R2: $X_i \longrightarrow X_j \relbar X_k$ $\Rightarrow$ $X_i \longrightarrow X_j \longrightarrow X_k$

R3: $X_i \overset{\frown}{\longrightarrow} X_j \longrightarrow X_k$ $\Rightarrow$ $X_i \overset{\frown}{\longrightarrow} X_j \longrightarrow X_k$

R4:

Structure Learning for BNs: Independence Test Based Approach

- In practice, we do not have access to p but to a finite sample from it. Then, replace $X_i \perp_p X_j | S$ in the PC algorithm with an independece test, preferably with one that is consistent so that the algorithm is **asymptotically** correct.

Structure Learning for BNs: Independence Test Based Approach

- In practice, we do not have access to p but to a finite sample from it. Then, replace $X_i \perp_p X_j | S$ in the PC algorithm with an indepedence test, preferably with one that is consistent so that the algorithm is **asymptotically** correct.

- Let $d_{1:N}$ be a complete sample. Then, $X_i \perp_p X_j | S$ implies that $p(X_i, X_j | S) = p(X_i | S) p(X_j | S)$ and thus that

$$N_{x_i, x_j, s} \approx N_{x_i, s} N_{x_j, s} / N$$

where $N_{x_i, x_j, s}$ is the number of instances in $d_{1:N}$ where x_i, x_j and s, and $N_{x_i, s} = \sum_{x_j} N_{x_i, x_j, s}$ and $N_{x_j, s} = \sum_{x_i} N_{x_i, x_j, s}$.

Structure Learning for BNs: Independence Test Based Approach

- In practice, we do not have access to p but to a finite sample from it. Then, replace $X_i \perp_p X_j | S$ in the PC algorithm with an indepedence test, preferably with one that is consistent so that the algorithm is **asymptotically** correct.

- Let $d_{1:N}$ be a complete sample. Then, $X_i \perp_p X_j | S$ implies that $p(X_i, X_j | S) = p(X_i | S) p(X_j | S)$ and thus that

$$N_{x_i, x_j, s} \approx N_{x_i, s} N_{x_j, s} / N$$

  where $N_{x_i, x_j, s}$ is the number of instances in $d_{1:N}$ where x_i, x_j and s, and $N_{x_i, s} = \sum_{x_j} N_{x_i, x_j, s}$ and $N_{x_j, s} = \sum_{x_i} N_{x_i, x_j, s}$.

- We can measure the deviance from the expected situation above by

$$deviance = \sum_{x_i, x_j, s} \frac{[N_{x_i, x_j, s} - N_{x_i, s} N_{x_j, s} / N]^2}{N_{x_i, s} N_{x_j, s} / N}$$

Structure Learning for BNs: Independence Test Based Approach

- In practice, we do not have access to p but to a finite sample from it. Then, replace $X_i \perp_p X_j | S$ in the PC algorithm with an independence test, preferably with one that is consistent so that the algorithm is **asymptotically** correct.

- Let $d_{1:N}$ be a complete sample. Then, $X_i \perp_p X_j | S$ implies that $p(X_i, X_j | S) = p(X_i | S) p(X_j | S)$ and thus that

$$N_{x_i, x_j, s} \approx N_{x_i, s} N_{x_j, s} / N$$

where $N_{x_i, x_j, s}$ is the number of instances in $d_{1:N}$ where x_i, x_j and s, and $N_{x_i, s} = \sum_{x_j} N_{x_i, x_j, s}$ and $N_{x_j, s} = \sum_{x_i} N_{x_i, x_j, s}$.

- We can measure the deviance from the expected situation above by

$$deviance = \sum_{x_i, x_j, s} \frac{[N_{x_i, x_j, s} - N_{x_i, s} N_{x_j, s} / N]^2}{N_{x_i, s} N_{x_j, s} / N}$$

- If the deviance is too large, then reject the hypothesis that $X_i \perp_p X_j | S$.

Structure Learning for BNs: Independence Test Based Approach

- In practice, we do not have access to p but to a finite sample from it. Then, replace $X_i \perp_p X_j | S$ in the PC algorithm with an independece test, preferably with one that is consistent so that the algorithm is **asymptotically** correct.

- Let $d_{1:N}$ be a complete sample. Then, $X_i \perp_p X_j | S$ implies that $p(X_i, X_j | S) = p(X_i | S) p(X_j | S)$ and thus that

$$N_{x_i, x_j, s} \approx N_{x_i, s} N_{x_j, s} / N$$

where $N_{x_i, x_j, s}$ is the number of instances in $d_{1:N}$ where x_i, x_j and s, and $N_{x_i, s} = \sum_{x_j} N_{x_i, x_j, s}$ and $N_{x_j, s} = \sum_{x_i} N_{x_i, x_j, s}$.

- We can measure the deviance from the expected situation above by

$$deviance = \sum_{x_i, x_j, s} \frac{[N_{x_i, x_j, s} - N_{x_i, s} N_{x_j, s} / N]^2}{N_{x_i, s} N_{x_j, s} / N}$$

- If the deviance is too large, then reject the hypothesis that $X_i \perp_p X_j | S$.

- Asymptotically, the deviance follows a χ^2 distribution with the appropriate number of degrees of freedom. Then, we can control the probability of falsely rejecting the hypothesis, a.k.a. p-value.

- **Exercise**. Run the PC algorithm assuming that p is faithful to the following DAGs.
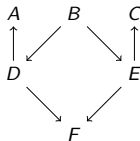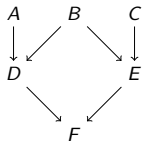
Structure Learning for BNs: Independence Test Based Approach

> ‣ Two DAGs represent the same independencies (i.e. they are **equivalent**) if and only if they have the same adjacencies and **unshielded colliders**, i.e. subgraphs $X_i \to X_k \leftarrow X_j$ where X_i and X_j are not adjacent.

Structure Learning for BNs: Independence Test Based Approach

- ▸ Two DAGs represent the same independencies (i.e. they are **equivalent**) if and only if they have the same adjacencies and **unshielded colliders**, i.e. subgraphs $X_i \to X_k \leftarrow X_j$ where X_i and X_j are not adjacent.
- ▸ The output of the PC algorithm is not a DAG in general, but an **essential graph** (EG):
  - ▸ H has an edge $X_i \to X_j$ if and only if $X_i \to X_j$ is in **every** DAG that is equivalent to G.

Structure Learning for BNs: Independence Test Based Approach

- Two DAGs represent the same independencies (i.e. they are **equivalent**) if and only if they have the same adjacencies and **unshielded colliders**, i.e. subgraphs $X_i \rightarrow X_k \leftarrow X_j$ where X_i and X_j are not adjacent.
- The output of the PC algorithm is not a DAG in general, but an **essential graph** (EG):
  - H has an edge $X_i \rightarrow X_j$ if and only if $X_i \rightarrow X_j$ is in **every** DAG that is equivalent to G.
  - In other words, H has an edge $X_i - X_j$ if and only if $X_i \rightarrow X_j$ is in some DAG that is equivalent to G and $X_i \leftarrow X_j$ is in some other DAG that is equivalent to G.

Structure Learning for BNs: Independence Test Based Approach

- ▸ Two DAGs represent the same independencies (i.e. they are **equivalent**) if and only if they have the same adjacencies and **unshielded colliders**, i.e. subgraphs $X_i \rightarrow X_k \leftarrow X_j$ where X_i and X_j are not adjacent.

- ▸ The output of the PC algorithm is not a DAG in general, but an **essential graph** (EG):
  - ▸ H has an edge $X_i \rightarrow X_j$ if and only if $X_i \rightarrow X_j$ is in **every** DAG that is equivalent to G.
  - ▸ In other words, H has an edge $X_i - X_j$ if and only if $X_i \rightarrow X_j$ is in some DAG that is equivalent to G and $X_i \leftarrow X_j$ is in some other DAG that is equivalent to G.

- ▸ A naive way to convert H into a DAG that is equivalent to G is as follows:

Repeat while possible
    Replace any edge $X_i - X_j$ in H with $X_i \rightarrow X_j$ if this does not create a directed cycle
    or a new unshielded collider
If H is not a DAG, then backtrack

Structure Learning for BNs: Score Based Approach

- Alternatively, we can choose the DAG G with maximum posterior probability (a.k.a **Bayesian score**):

$$p(G|d_{1:N}) = p(d_{1:N}|G)p(G)/P(d_{1:N}) \propto p(d_{1:N}|G)p(G)$$

where $p(d_{1:N}|G)$ is the marginal likelihood of $d_{1:N}$ given G, $p(G)$ is a prior probability distribution, and $p(d_{1:N})$ is a normalization constant.

Structure Learning for BNs: Score Based Approach

▸ Alternatively, we can choose the DAG G with maximum posterior probability (a.k.a **Bayesian score**):

$$p(G|d_{1:N}) = p(d_{1:N}|G)p(G)/P(d_{1:N}) \propto p(d_{1:N}|G)p(G)$$

where $p(d_{1:N}|G)$ is the marginal likelihood of $d_{1:N}$ given G, $p(G)$ is a prior probability distribution, and $p(d_{1:N})$ is a normalization constant.

▸ Moreover

$$p(d_{1:N}|G) = \int p(d_{1:N}|\theta_G, G)p(\theta_G|G)d\theta_G$$

where $p(d_{1:N}|\theta_G, G)$ is the likelihood function of $d_{1:N}$ given G and θ_G, and $p(\theta_G|G)$ is a prior probability distribution.

Structure Learning for BNs: Score Based Approach

▸ Alternatively, we can choose the DAG G with maximum posterior probability (a.k.a **Bayesian score**):

$$p(G|d_{1:N}) = p(d_{1:N}|G)p(G)/P(d_{1:N}) \propto p(d_{1:N}|G)p(G)$$

where $p(d_{1:N}|G)$ is the marginal likelihood of $d_{1:N}$ given G, $p(G)$ is a prior probability distribution, and $p(d_{1:N})$ is a normalization constant.

▸ Moreover

$$p(d_{1:N}|G) = \int p(d_{1:N}|\theta_G, G)p(\theta_G|G)d\theta_G$$

where $p(d_{1:N}|\theta_G, G)$ is the likelihood function of $d_{1:N}$ given G and θ_G, and $p(\theta_G|G)$ is a prior probability distribution.

▸ **Assuming** that $p(\theta_G|G) = \prod_i \prod_j p(\theta_{X_i|pa_G(X_i)=j}|G)$ and $p(\theta_{X_i|pa_G(X_i)=j}|G) \sim Dirichlet(\alpha_{ij1}, \dots, \alpha_{ijk_i})$, we have that

$$p(d_{1:N}|G) = \prod_i \prod_j \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_k \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}$$

where $\alpha_{ij} = \sum_k \alpha_{ijk}$, N_{ijk} is the number of instances in $d_{1:N}$ where $X_i = k$ and $pa_G(X_i) = j$, and $N_{ij} = \sum_k N_{ijk}$.

Structure Learning for BNs: Score Based Approach
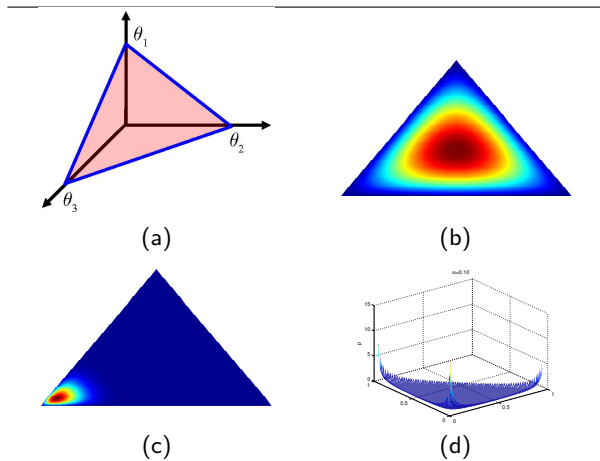
(a) The Dirichlet distribution over a 3-valued random variable is defined over the simplex represented by the triangular surface. Points in this surface satisfy $0 \le \theta_i \le 1$ and $\sum_i \theta_i = 1$. (b) Dirichlet(2,2,2). (c) Dirichlet(20,2,2). (d) Dirichlet(0.1,0.1,0.1). Source: Murphy (2012).

Structure Learning for BNs: Score Based Approach

▸ The Bayesian score is **score equivalent** (i.e. it gives the same score to equivalent DAGs) if and only if

$$\alpha_{ijk} = \alpha p'(ijk)$$

where α is the user-defined imaginary sample size (the higher the less regularization) and $p'(ijk)$ is a prior probability distribution. For instance, $p'(ijk) = 1/[k_i \prod_{X_l \in pa_G(X_i)} k_l]$ results in the so-called BDeu score.

Structure Learning for BNs: Score Based Approach

- The Bayesian score is **score equivalent** (i.e. it gives the same score to equivalent DAGs) if and only if

$$\alpha_{ijk} = \alpha p'(ijk)$$

where α is the user-defined imaginary sample size (the higher the less regularization) and $p'(ijk)$ is a prior probability distribution. For instance, $p'(ijk) = 1/[k_i \prod_{X_l \in pa_G(X_i)} k_l]$ results in the so-called BDeu score.

- Under the Dirichlet parameter prior assumption and when $N \to \infty$, we have that

$$\log p(d_{1:N}|G) \approx \log p(d_{1:N}|\theta_G^{ML}, G) - \frac{\log N}{2} dim(G)$$

where $dim(G)$ is the dimension or number of free parameters of G, i.e. $\sum_i (k_i - 1) \prod_{X_l \in pa_G(X_i)} k_l$.

Structure Learning for BNs: Score Based Approach

- ▸ The Bayesian score is **score equivalent** (i.e. it gives the same score to equivalent DAGs) if and only if

$$\alpha_{ijk} = \alpha p'(ijk)$$

where α is the user-defined imaginary sample size (the higher the less regularization) and $p'(ijk)$ is a prior probability distribution. For instance, $p'(ijk) = 1/[k_i \prod_{X_l \in pa_G(X_i)} k_l]$ results in the so-called BDeu score.

- ▸ Under the Dirichlet parameter prior assumption and when $N \to \infty$, we have that

$$\log p(d_{1:N}|G) \approx \log p(d_{1:N}|\theta_G^{ML}, G) - \frac{\log N}{2} dim(G)$$

where $dim(G)$ is the dimension or number of free parameters of G, i.e. $\sum_i (k_i - 1) \prod_{X_l \in pa_G(X_i)} k_l$.

- ▸ This approximation is called **Bayesian information criterion** (BIC), and it shows that the Bayesian score favours models that trade off fit of data and model complexity.

Structure Learning for BNs: Score Based Approach

- Number of DAGs with 1-12 nodes: 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143, 31603459396418917607425, 521939651343829405020504063

Structure Learning for BNs: Score Based Approach

- Number of DAGs with 1-12 nodes: 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143, 31603459396418917607425, 521939651343829405020504063
- Then, an exhaustive search is prohibitive. Then, a heuristic search must be performed instead.

Hill-climbing (HC)

Let G be the empty DAG
Repeat until no change occurs
    Add, remove or reverse any edge in G that improves the Bayesian score the most

---

Structure Learning for BNs: Score Based Approach

- Number of DAGs with 1-12 nodes: 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143, 31603459396418917607425, 521939651343829405020504063

- Then, an exhaustive search is prohibitive. Then, a heuristic search must be performed instead.

Hill-climbing (HC)

Let G be the empty DAG
Repeat until no change occurs
    Add, remove or reverse any edge in G that improves the Bayesian score the most

---

- The log Bayesian score is **decomposable** if $\log p(G)$ is so. That is

$$\log p(G|d_{1:N}) = \sum_i f(X_i, pa_G(X_i), d_{1:N})$$

Structure Learning for BNs: Score Based Approach

▸ Number of DAGs with 1-12 nodes: 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143, 31603459396418917607425, 521939651343829405020504063

▸ Then, an exhaustive search is prohibitive. Then, a heuristic search must be performed instead.

Hill-climbing (HC)

Let G be the empty DAG
Repeat until no change occurs
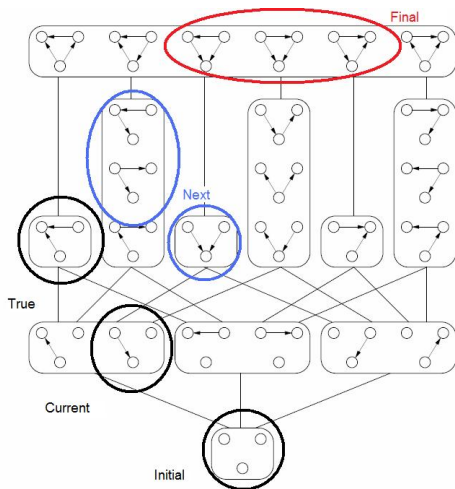   Add, remove or reverse any edge in G that improves the Bayesian score the most

---

▸ The log Bayesian score is **decomposable** if $\log p(G)$ is so. That is

$$\log p(G|d_{1:N}) = \sum_i f(X_i, pa_G(X_i), d_{1:N})$$

▸ Then, adding, removing or reversing a edge in G implies recomputing only one or two factors.

Structure Learning for BNs: Score Based Approach

- Unfortunately, HC is not asymptotically correct.

Structure Learning for MNs: Independence Test Based Approach

▸ We can get an UG H such that $p(X) = \prod_i \varphi(C_i)/Z$ and, thus, that we can use for probabilistic reasoning as follows:

For each X_i do
    Set $ad_H(X_i)$ to be any minimal subset of $X \smallsetminus X_i$ such that
    $X_i \perp_p X \smallsetminus ad_H(X_i) | ad_H(X_i)$

Structure Learning for MNs: Independence Test Based Approach

- We can get an UG H such that $p(X) = \prod_i \varphi(C_i)/Z$ and, thus, that we can use for probabilistic reasoning as follows:

  For each X_i do
      Set $ad_H(X_i)$ to be any minimal subset of $X \smallsetminus X_i$ such that
      $X_i \perp_p X \smallsetminus ad_H(X_i) | ad_H(X_i)$

- Luckily, we can get H without searching over the 2^{n-1} possible adjacent sets for each node if we assume **faithfulness**, i.e. $U \perp_p V | Z$ if and only if $U \perp_G V | Z$ for some UG G.

  Incremental associative Markov boundary algorithm (IAMB)

  For each X_i do
      $ad_H(X_i) := \varnothing$
      Repeat until no change occurs
          if there exists $X_j \notin ad_H(X_i) \cup X_i$ such that $X_i \not\perp_p X_j | ad_H(X_i)$ then
              $ad_H(X_i) := ad_H(X_i) \cup X_j$
      Repeat until no change occurs
          if there exists $X_j \in ad_H(X_i)$ such that $X_i \perp_p X_j | ad_H(X_i) \smallsetminus X_j$ then
              $ad_H(X_i) := ad_H(X_i) \smallsetminus X_j$

Structure Learning for MNs: Score Based Approach

- ▸ **Exercise**. Sketch how to perform structure learning for MNs. Consider issues such as score decomposability, existence of closed form expressions, and problems due to equivalent MNs.
- ▸ **Exercise**. Sketch how to perform structure learning for BNs and MNs from an incomplete sample. Consider issues such as score decomposability, existence of closed form expressions, and problems due to equivalent BNs and MNs.

Contents

- Structure Learning for BNs
  - Independence Test Based Approach
  - Score Based Approach
- Structure Learning for MNs
  - Independence Test Based Approach

Contents

- Structure Learning for BNs
  - Independence Test Based Approach
  - Score Based Approach
- Structure Learning for MNs
  - Independence Test Based Approach

Thank you