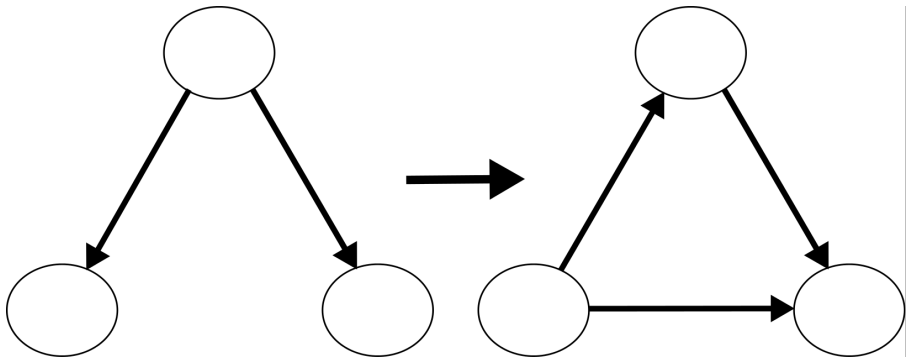
Bayesian networks: learning

Probabilistic Graphical Models

L. Enrique Sucar, INAOE



Outline

Introduction

- Learning a Bayesian network includes two aspects:
  - Structure Learning. There are two main types of methods: (i) global methods based on search and score, and (ii) local methods that use conditional independence tests
  - Parameter Learning. When the structure is known, parameter learning consists in estimating the conditional probability tables (CPTs) from data.

Parameter Learning

- If we have *sufficient* and complete data for all the variables, and we assume the topology of the BN is known, parameter learning is straight forward
- The CPT for each variable can be estimated from the data based on the frequency of each value obtaining a *maximum likelihood* (ML) estimator
- For example, to estimate the CPT of B given it has two parents, A, C:

$$P(B_i \mid A_j, C_k) \sim NB_iA_jC_k / NA_jC_k \tag{1}$$

Smoothing

- When we estimate probabilities from data, it can sometimes happen that a particular event never occurs – the corresponding probability value is zero, implying an *impossible* case
- The previous situation can be avoided by using some type of *smoothing* for the probabilities, eliminating zero probability values
- There are several smoothing techniques, one of the most common and simplest is *Laplacian* smoothing

Laplacian smoothing

- Laplacian smoothing consists in initializing the probabilities to a uniform distribution, and then updating these values based on the data
- Consider a discrete variable, X, with k possible values. Initially, each probability will be set to $P(x_i) = 1/k$
- Data set with N samples, in which the value x_i occurs m times; the estimate of its probability will be the following:

$$P(x_i) = (1 + m)/(k + N) \tag{2}$$

Parameter uncertainty

- If there is not sufficient data we have uncertainty in the parameters
- This uncertainty can be modelled using a second order probability distribution
- For binary variables, the uncertainty in the parameters can be modelled using a Beta distribution:

$$\beta(a,b) = \frac{(a+b+1)!}{a!b!} x^a (1-x)^b \tag{3}$$

- For multivalued variables, uncertainty in the parameters can be represented by the Dirichlet distribution

Representing expert's uncertainty

- For the binary case, the expected value of the Beta distribution is given by: $P(b_i) = a + 1/a + b + 2$, where a and b are the parameters of the Beta distribution
- The parameters of the Beta distribution can represent a measure of *confidence* in the expert's estimates, expressed by varying the term $a + b$:
  - Complete ignorance: $a = b = 0$.
  - Low confidence: $a + b$ *small* (10).
  - Medium confidence: $a + b$ *intermediate* (100).
  - High confidence: $a + b$ large (1000).

Combining expert estimate and data

- This representation could be used to combine experts' estimations with data
- To approximate the probability value of a binary variable, b_i we can use:

$$P(b_i) = k + a + 1/n + a + b + 2 \tag{4}$$

Where $a/a + b$ represents the expert's estimate, and k/n is the probability obtained from the data

Example

- Assume an expert gives an estimate of 0.7 for a certain parameter, and that the experimental data provides 40 positive cases among 100 samples:

Low confidence ($a + b = 10$):
$P(b_i) = \frac{40+7+1}{100+10+2} = 0.43$
Medium confidence ($a + b = 100$):
$P(b_i) = \frac{40+70+1}{100+100+2} = 0.55$
High confidence ($a + b = 1000$):
$P(b_i) = \frac{40+700+1}{100+1000+2} = 0.67$

Incomplete data

- Another common situation is to have incomplete data:

  Missing values: In some registers there are missing values for one or more variables.

  Hidden nodes: A variable or set of variables in the model for which there is no data at all.

Missing values

- There are several alternatives:
  1. Eliminate the registers with missing values.
  2. Consider a special “unknown” value.
  3. Substitute the missing value by the most common value (mode) of the variable.
  4. Estimate the missing value based on the values of the other variables in the corresponding register.
- In general the best alternative is the fourth option

Estimating missing values

- Learn the parameters of the BN based on the complete registers, and then complete the data and re-estimate the parameters:
  1. Instantiate all the known variables in the register.
  2. Through probabilistic inference obtain the posterior probabilities of the missing variables.
  3. Assign to each variable the value with highest posterior probability.
  4. Add this completed register to the database and re-estimate the parameters.
- An alternative is to assign a *partial* case for each value of the variable proportional to the posterior probability.

Hidden nodes

- An approach to estimate their parameters is based on the *Expectation–Maximization* (EM) technique
- It consists of two phases which are repeated iteratively:
  - E step: the missing data values are estimated based on the current parameters.
  - M step: the parameters are updated based on the estimated data.
- The algorithm starts by initializing the missing parameters with random values

Algorithm

1. Obtain the CPTs for all the *complete* variables (the values of the variable and all its parents are in the database) based on a ML estimator.
2. Initialize the unknown parameters with random values.
3. Considering the actual parameters, estimate the values of the hidden nodes based on the known variables via probabilistic inference.
4. Use the estimated values for the hidden nodes to complete/update the database.
5. Re-estimate the parameters for the hidden nodes with the updated data.
6. Repeat 3–5 until converge (no significant changes in the parameters).

Example - golf data

Outlook	Temperature 1	Humidity	Wind	Play
sunny	xxx	high	–	N
sunny	high	high	–	N
overcast	high	high	–	P
rainy	medium	high	–	P
rainy	low	normal	–	P
rainy	low	normal	–	N
overcast	low	normal	–	P
sunny	medium	high	–	N
sunny	xxx	normal	–	P
rainy	medium	normal	–	P
sunny	medium	normal	–	P
overcast	medium	high	–	P
overcast	high	normal	–	P
rainy	medium	high	–	N

Example

- Assume that we learn a naive Bayes classifier considering *Play* as the class variable
- Missing values – estimate the probability of temperature for the registers in which it is missing, via probabilistic inference:

  Register 1: $P(Temperature \mid sunny, high, N)$
  Register 9: $P(Temperature \mid sunny, normal, P)$
- For the case of the hidden node, *Wind*, we cannot obtain the corresponding CPT, $P(Wind \mid Play)$
- Apply the EM procedure, first pose initial random parameters for the CPT:

$$P(Wind \mid Play) = \begin{matrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{matrix}$$

Example

- Given this CPT we have a complete initial model for the NBC, and can estimate the probability of wind for each register based on the values of the other variables in the register
- By selecting the highest probability value for each register, we can fill-in the table
- Based on this new data table, we re-estimate the parameters, and obtain a new CPT:

$$P(Wind \mid Play) = \begin{matrix} 0.60 & 0.44 \\ 0.40 & 0.56 \end{matrix}$$

- The process is then repeated until he EM procedure has converged

Example - completed data set

Outlook	Temperature 1	Humidity	Wind	Play
sunny	medium	high	no	N
sunny	high	high	no	N
overcast	high	high	no	P
rainy	medium	high	no	P
rainy	low	normal	yes	P
rainy	low	normal	yes	N
overcast	low	normal	yes	P
sunny	medium	high	no	N
sunny	medium	normal	no	P
rainy	medium	normal	no	P
sunny	medium	normal	yes	P
overcast	medium	high	yes	P
overcast	high	normal	yes	P
rainy	medium	high	yes	N

Discretization

- Usually Bayesian networks consider discrete or nominal values.
- An alternative to include continuous variables in BNs is to discretize them.
- Discretization methods can be (i) unsupervised and (ii) supervised.

Unsupervised Discretization

- The two main types of unsupervised discretization approaches are: equal width and equal data.
- Equal width consists in dividing the range of a variable, $[Xmin; Xmax]$, in k equal bins; such that each bin has a size of $[Xmin; Xmax]/k$
- Equal data divides the range of the variable in k intervals, such that each interval includes the same number of data points from the training data

Supervised Discretization

- Supervised discretization considers the task to be performed with the model, such that the variables are discretized to optimize this task, for instance classification accuracy
- If we consider a BN for classification with continuous attribute variables, these are discretized according to the class values
- Consider the attribute variable X with range $[Xmin; Xmax]$ – he problem is to determine the *optimal* partition of X such that the classifier precision is maximized

Algorithm

1. Generate all potential divisions in X which correspond to a value in $[Xmin; Xmax]$ where there is a change in the class value.
2. Based on the potential division points generate an initial set of n intervals.
3. Test the classification accuracy of the Bayesian classifier (usually on a different set of data known as a validation set) according to the current discretization.
4. Modify the discretization by partitioning an interval or joining two intervals.
5. Repeat (3) and (4) until the accuracy of the classifier cannot be improved or some other termination criteria occurs.

Structure learning

- Structure learning consists in obtaining the topology of the BN from the data
- This is a complex problem because: (i) the number of possible structures is *huge* even with a few variables (it is super-exponential on the number of variables); (ii) a very large database is required to obtain good estimates of the statistical measures
- There are several techniques depending on the type of structure – trees, polytrees, general DAG

Tree learning - Chow and Liu algorithm

- The joint probability of n random variables can be approximated as:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid X_{j(i)}) \tag{5}$$

  where $X_{j(i)}$ is the parent of X_i in the tree.
- The problem consists in obtaining the *best* tree
- A measure of how close the approximation is based on the information difference between the real distribution (P) and the tree approximation (P^*) is as follows:

$$DI(P, P^*) = \sum_{X} P(X) log(P(X)/P^*(X)) \tag{6}$$

Tree learning

- The mutual information between any pair of variables is defined as:

$$I(X_i, X_j) = \sum_{X_i, X_j} P(X_i, X_j) log(P(X_i, X_j)/P(X_i)P(X_j)) \quad (7)$$

- Given a tree-structured BN with variables $X_1, X_2, ..., X_n$, we define its *weight*, W, as the sum of the mutual information of the arcs:

$$W(X_1, X_2, ..., X_n) = \sum_{i=1}^{n-1} I(X_i, X_j) \quad (8)$$

- It can be shown that minimizing DI is equivalent to maximizing W

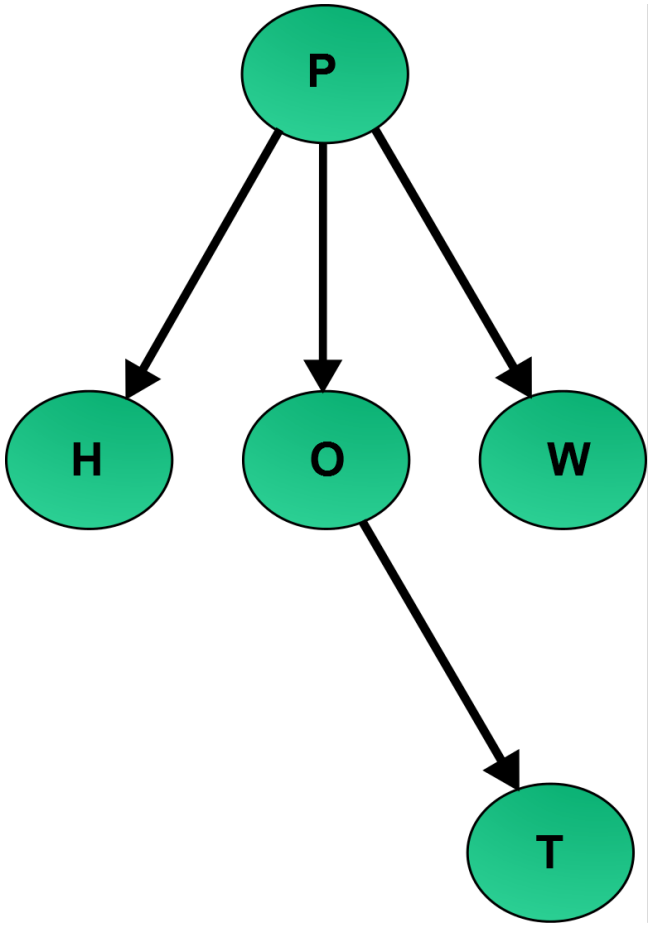
Algorithm - *maximum weight spanning tree*

1. Obtain the mutual information (I) between all pairs of variables (for n variables, there are $n(n-1)/2$ pairs).
2. Order the mutual information values in descending order.
3. Select the pair with maximum I and connect the two variables with an arc, this constitutes the initial tree
4. Add the pair with the next highest I to the tree, while they do not make a cycle; otherwise skip it and continue with the following pair.
5. Repeat 4 until all the variables are in the tree ($n-1$ arcs).

Example - mutual information for golf

No.	Var 1	Var 2	Mutual Info.
1	temp.	outlook	.2856
2	play	outlook	.0743
3	play	humidity	.0456
4	play	wind	.0074
5	humidity	outlook	.0060
6	wind	temp.	.0052
7	wind	outlook	.0017
8	play	temp.	.0003
9	humidity	temp.	0
10	wind	humidity	0

Example - tree



Learning polytrees

- Chow and Liu algorithm obtains only the *skeleton* of the tree
- Rebane and Pearl developed a method that can be used to direct the arcs in the skeleton
- The algorithm is based on independence tests for variable triplets, and in this way it can distinguish *convergent* substructures
- Once one or more substructures of this type are detected in the skeleton, it can direct additional arcs by applying the independence tests to neighboring nodes

Independence tests

- Given three variables, there are three possibilities:
  1. Sequential arcs: $X \rightarrow Y \rightarrow Z$.
  2. Divergent arcs: $X \leftarrow Y \rightarrow Z$.
  3. Convergent arcs: $X \rightarrow Y \leftarrow Z$.
- The first two cases are indistinguishable; however the third case is different, since X and Z are NOT independent given Y
- So this case can be used to determine the directions of the two arcs that connect these three variables

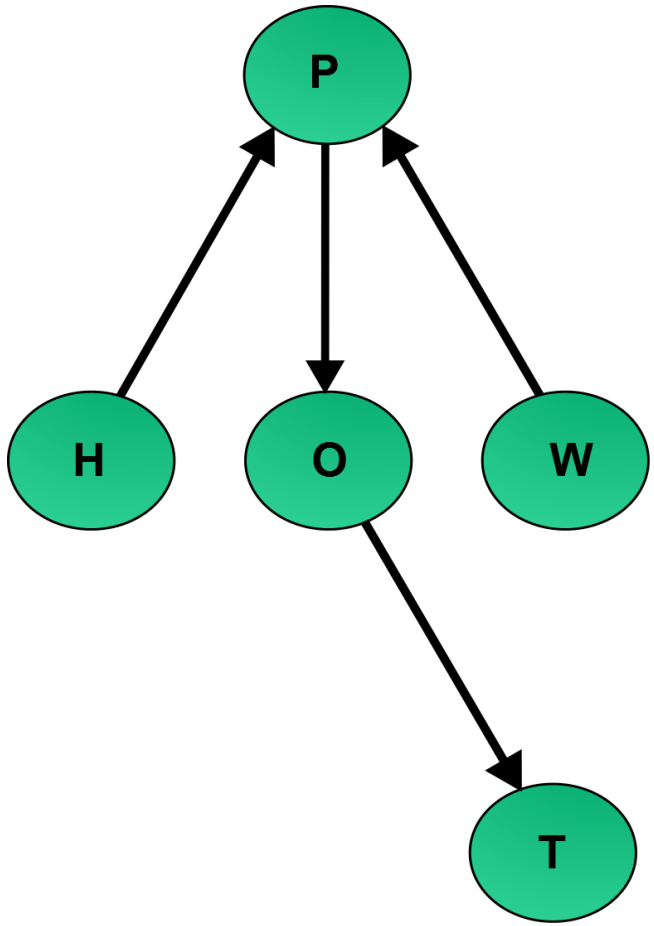
Algorithm

1. Obtain the skeleton using the Chow and Liu algorithm.
2. Iterate over the network until a convergent variable triplet is found. We will call the variable to which the arcs converge a *multi-parent node*.
3. Starting with a multi-parent node, determine the directions of other arcs using independence tests for variable triplets. Continue this procedure until it is no longer possible (causal base).
4. Repeat 2-3 until no other directions can be determined.
5. If any arcs are left undirected, use the external semantics to infer their directions.

Example - golf

- Variable triplet H, P, W falls in the convergent case. Then, the arcs will be directed such that H points to P and W points to P
- If H and W are independent from O given P then there will be an arc that points from P to O
- Finally, the dependence relation between P and T given O is tested, and if they are again found to be independent, then the arc points from O to T

Example - golf polytree



General case

- For the general case several methods have been proposed, which can be divided into two main classes:
  1. Global methods: these perform a heuristic search over the space of network structures, starting from some initial structure, and generating a variation of the structure at each step. The *best* structure is selected based on a score that measures how well the model represents the data. Common scores are BIC and MDL
  2. Local methods: these are based on evaluating the (in)dependence relations between subsets of variables given the data, to sequentially obtain the structure of the network. The most well known variant of this approach is the PC algorithm

Global methods

- Global methods search for the *best* structure based on a global metric
- Different structures are generated and these are evaluated with respect to the data using some scoring method
- There are several variants that depend on two aspects: (i) a fitness measure between the structure and the data, and (ii) a method for searching for the best structure

Scoring functions

- A scoring function evaluates how well a structure fits the data
- Common scoring functions are: the maximum likelihood (ML), the Bayesian information criterion (BIC), the Bayesian score (BD), and the minimum description length (MDL) criterion
- The score must balance the precision and complexity of the model

K2 Metric

- This score is decomposable and it is calculated for each variable X_i given its parents $Pa(X_i)$:

$$S_i = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \tag{9}$$

- Where r_i is the number of values of X_i, q_i is the number of possible configurations for the parents of X_i, α_{ijk} is the number of cases in the database where $X_i = k$ and $Pa(X_i) = j$, and N_{ij} is the number of cases in the database where $Pa(X_i) = j$

MDL

- The MDL measure makes a compromise between accuracy and model complexity:

$$MC = \alpha(W/Wmax) + (1-\alpha)(1 - L/Lmax) \quad (10)$$

where W represents the accuracy of the model, and L the complexity. $Wmax$ and $Lmax$ represent the maximum accuracy and complexity

- Complexity is given by the number of parameters required for representing the model:

$$L = S_i[k_i log_2 n + d(S_i - 1)F_i] \quad (11)$$

where n is the number of nodes, k is the number of parents per node, S_i is the average number of values per variable, F_i is the average number of values per parent variable, and d the number of bits per parameter

MDL

- The accuracy can be estimated based on the 'weight' of each node:

$$w(X_i, Pa(X_i)) =$$

$$\sum_{xi} P(X_i, Pa(X_i)) log[P(X_i, Pa(X_i))/P(X_i)P(Pa(X_i))] \quad (12)$$

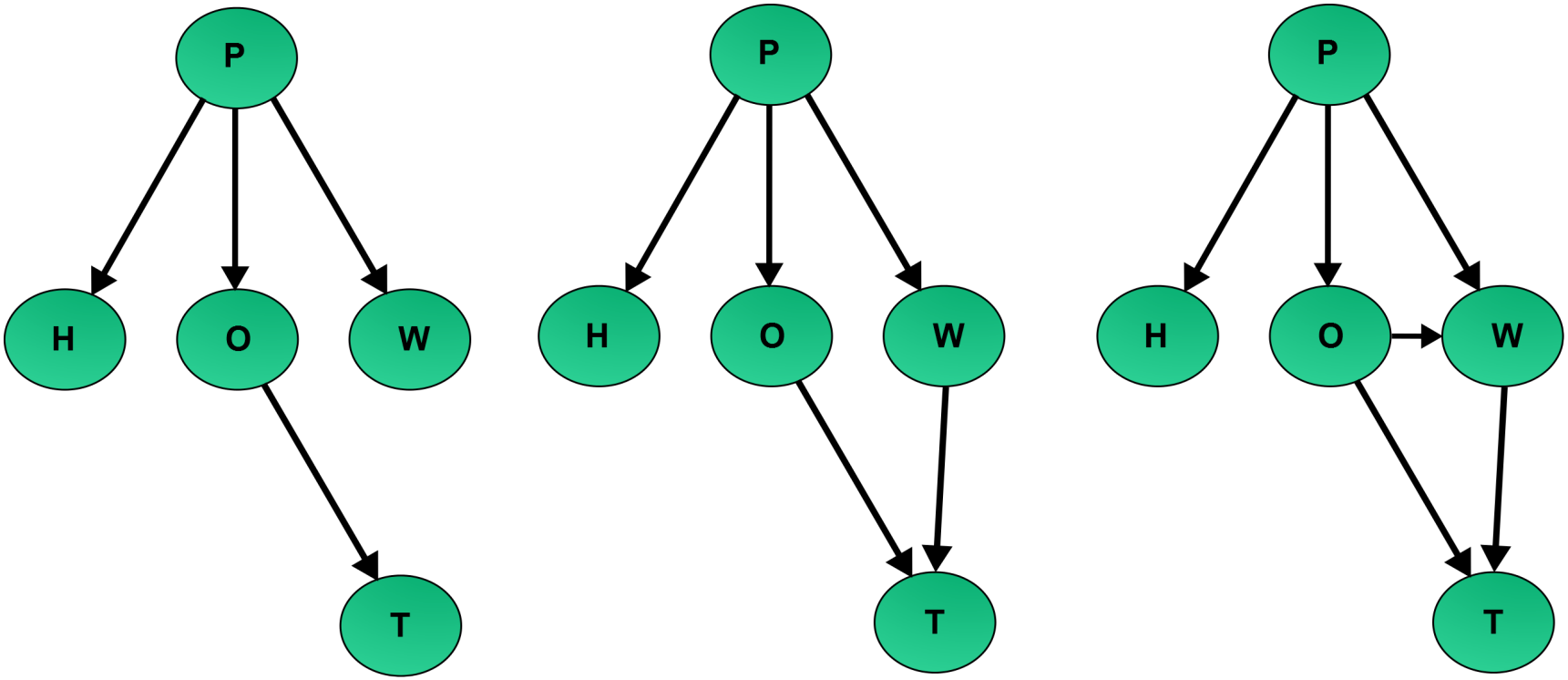
and the weight (accuracy) total is given by the sum of the weights for each node:

$$W = \sum_i w(X_i, Pa(X_i)) \quad (13)$$

Search Algorithms

- Since the number of possible structures is exponential on the number of variables, heuristic approaches are used for searching for the "best" structure
- One common strategy is to use *hill climbing*:
  1. Generate an initial structure - tree
  2. Calculate the fitness measure of the initial structure.
  3. Add/ invert an arc from the current structure.
  4. Calculate the fitness measure of the new structure.
  5. If the fitness improves, keep the change; if not, return to the previous structure.
  6. Repeat 3 -5 until no further improvements exist.
- A common global method is K2 (see text for details)

Example - global methods



Local Methods

- The basic idea is to apply independence tests to sets of variables to recover the structure of the BN
- A common local technique is the PC algorithm
- The PC algorithm first recovers the skeleton (underlying undirected graph) of the BN, and then it determines the orientation of the edges

PC

- To determine the skeleton, it starts from a fully connected undirected graph, and determines the conditional independence of each pair of variables given some subset of the other variables - using the conditional cross entropy measure
- In the second phase the direction of the edges are set based on conditional independence tests between variable triplets

Algorithm

1. Initialize a complete undirected graph G'
2. $i = 0$
3. For $X \in \mathbf{X}$
4. For $Y \in ADJ(X)$
5. For $S \subseteq ADJ(X) - \{Y\}, \mid S \mid = i$
   - If $I(X, Y \mid S)$: Remove the edge $X - Y$ from G'
6. $i = i + 1$
7. Until $\mid ADJ(X) \mid \leq i, \forall X$
8. Orient edges in G'
9. Return G

Combining Experts and Data

- When domain expertise is available, this can be combined with learning algorithms to improve the model
- For structure learning, there are two basic approaches to combine expert knowledge and data:
  - Use expert knowledge as *restrictions* to reduce the search space for the learning algorithm.
  - Start from a structure proposed by an expert and use data to validate and improve this structure.

Incorporating expert knowledge

- There are several ways to use expert knowledge to aid the structure learning algorithm, such as:
  1. Define an ordering for the variables (causal order), such that there could be an arc from X_i to X_j only if X_j is after X_i according to the specified ordering.
  2. Define restrictions in terms of directed arcs that must exist between two variables, i.e. $X_i \rightarrow X_j$.
  3. Define restrictions in terms of an arc between two variables that could be directed either way.
  4. Define restrictions in terms of pairs of variables that are not directly related, that is, there must be no arc between X_i and X_j.
  5. Combinations of the previous restrictions.
- In the case of the second approach, the structural improvement algorithm can be extended to general BN structures, in particular for tree-structured BNs

Applications

- There are many domains in which learning Bayesian networks has been applied to get a better understanding of the domain or make predictions based on partial observations; for example medicine, finance, industry and the environment, among others
- Next we present an example for modeling the air pollution in Mexico City

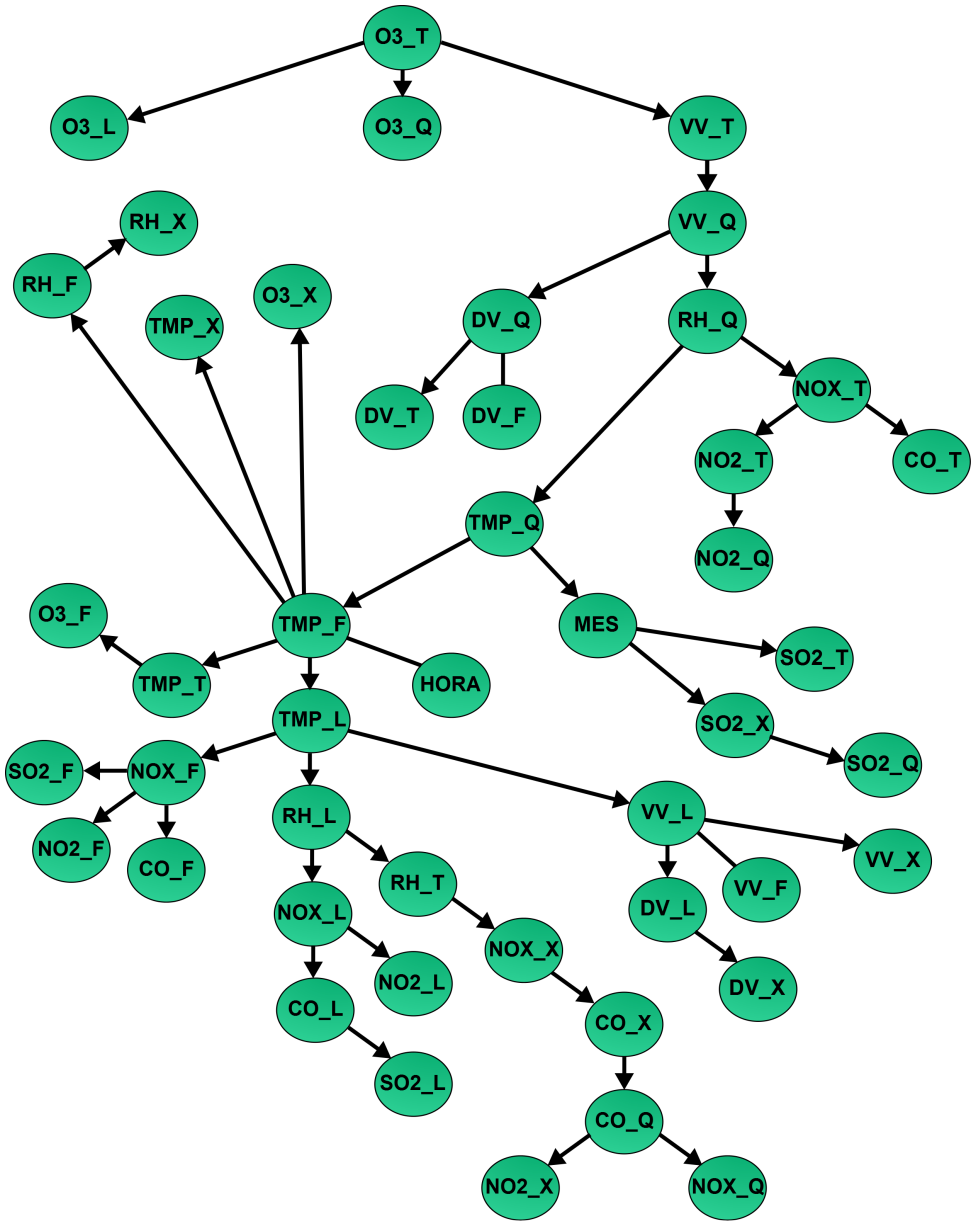
Air pollution model for Mexico City

- In Mexico City, the ozone level is used as a global indicator for the air quality. The concentrations of ozone are given in IMECA. It is important to predict the ozone level to take emergency measures if the pollution level is going to be above a certain threshold
- It is useful to know the dependencies between the different variables that are measured:
  - Determine which factors are more important for the ozone concentration in Mexico City.
  - Simplify the estimation problem, by taking into account only the relevant information.
  - Discover the most critical primary causes of pollution in Mexico City; these could help in future plans to reduce pollution.

Learning a BN for Air Pollution

- Apply a learning algorithm to obtain an initial structure of the phenomena
- 47 variables: 9 measurements for each of the 5 stations, plus the hour and month in which they were recorded
- We used nearly 400 random samples, and applied the Chow and Liu algorithm to obtain the tree structure that best approximates the data distribution

Model



Analysis

- From this initial structure we can get an idea of the relevance or influence of the other variables for estimating *ozone-Pedregal*. The nodes "closest" to the root are the most important ones, and the "far-away" nodes are less important.
- In this case we observe that there are 3 variables (ozone-Merced, ozone-Xalostoc, and wind velocity in Pedregal) that have the greatest influence in *ozone-Pedregal*
- We estimated *ozone-Pedregal* using only these 3 variables – the average error (absolute difference between the real and the estimated ozone concentration) is 11 IMECA or 12%, and for non-training data it is 26 IMECA or 22%

Book

Sucar, L. E, *Probabilistic Graphical Models*, Springer 2015 – Chapter 8

Additional Reading (1)


Chow, C.K., Liu, C.N.: Approximating Discrete Probability Distributions with Dependence Trees. IEEE Transactions on Information Theory. 14, 462–467 (1968)


Cooper, G.F., Herskovitz, E.: A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning. 9(4), 309–348 (1992)


Heckerman, D.: A Tutorial on Learning with Bayesian Networks. Innovations in Bayesian Networks. Springer Netherlands, 33–82 (2008)


Lam, W., Bacchus, F.: Learning Bayesian Belief Networks: An Approach based on the MDL Principle. Computational Intelligence. 10, 269–293 (1994)

Additional Reading (1)

- Neapolitan, R.E.: Learning Bayesian Networks. Prentice Hall, New Jersey (2004)
- Rebane G., Pearl, J.: The Recovery of Causal Poly-Trees from Statistical Data. In: Laveen N. Kanal, Tod S. Levitt, and John F. Lemmer (eds.) Uncertainty in Artificial Intelligence, pp. 175–182, (1987)
- Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. Springer-Verlag, Berlin (1993)
- Sucar, L.E., Ruiz-Suarez J. C.: Forecasting Air Pollution with Causal Probabilistic Networks. In: V. Barnett, K.F. Turkman (eds.)Statistics for the Environment 3: Statistical Aspects of Pollution, pp. 185–197. J. Wiley & Sons, Chichester (2007)