Imperial College London

NOTES

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

493 Data Analysis and Probabilistic Inference

Author:
(CID:)

Date: February 11, 2017

1 Bayes Theorem and Bayesian Inference

1. Bayes Theorem:

$$P(D \cap S) = P(D|S)P(S) = P(S|D)P(D)$$
$$P(D|S) = \alpha \times P(D) \times P(S|D)$$

2. Law of Total Probability:

$$P(F) = \sum_{i=1}^{n} P(F \cap E_i) = \sum_{i=1}^{n} P(F|E_i)P(E_i)$$

3. Conditional Independence: Two events D and S are conditionally independent given G if $P(G) \neq 0$ and one of the following holds:
   (a) $P(D|S \cap G) = P(D|G)$ and $P(D|G) \neq 0, P(S|G) \neq 0$
   (b) $P(D|G) = 0$ or $P(S|G) = 0$
   (c) $P(D \cap S|G) = P(D|G)P(S|G)$

4. Bayesian Inference: Given a set of competing hypothesis which explain a data set, for each hypothesis:
   (a) Convert the prior and likelihood information in the data into probabilities and take their product
   (b) Normalize the result to get the posterior probabilities of each hypothesis given the evidence
   (c) Select the most probably hypothesis

2 Simple Bayesian Networks

1. We have to assume the Causal Markov Condition has the following difficulties inherent in large instances

   (a) The joint probabilities are hard to estimates

   (b) Even if the joint probabilities can be obtained, there are too many number of instances

2. **Causal Markov Condition:** Suppose we have a joint probability distribution P of the random variables in some set $\mathcal{V}$ and a DAG $\mathbb{G} = (\mathcal{V}, E)$. We say that $(\mathbb{G}, P)$ satisfies the Markov condition if for each variable $X \in \mathcal{V}$, $\{X\}$ is conditional independent of **the set of all its nondescendents given the set of all its parents**. Let ND_X be the non-descendents and PA_X be the parents of X.

$$I_P(\{X\}, ND_X | PA_X)$$

3. Possible violations of the Causal Markov Condition:

   (a) Hidden cause: X and Y is said to have a common cause if there exists some variable that has causal paths into both of them. If we fail to model this common cause, in short (exists a hidden cause), the Markov condition would be violated as it assumes independence.

   (b) Selection bias: It is similar to hidden cause. The variables we observe shows independence when actually because of our sampling error.

   (c) Feedback loops: Causal relationships need to be only one way. The child node under no circumstances should influence the parent node.

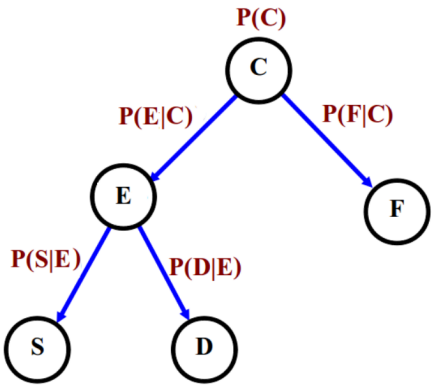


Figure 1: Example of naive bayes network. Given the parent C, the node E and F meet the causal markov condition, i.e. they are conditionally independent.

4. Each arc in a simple network is represented by a link matrix (conditional probability matrix)

$$\boldsymbol{P}(\boldsymbol{D}|\boldsymbol{C}) = \Big[P(d_i|c_j)\Big] = \begin{bmatrix} P(d_1|c_1) & P(d_1|c_2) \\ P(d_2|c_1) & P(d_2|c_2) \\ P(d_3|c_1) & P(d_3|c_2) \\ P(d_4|c_1) & P(d_4|c_2) \end{bmatrix}$$

5. The root nodes of a network do not have any parents. They have a vector giving the prior probabilities

$$\boldsymbol{P}(\boldsymbol{C}) = [P(c_i)] = \Big[P(c_1) \quad P(c_2)\Big]$$

6. **Instantiation** means setting the value of a node.

7. **Bayesian Classifiers** using the above network.

   (a) We cannot compute E as it is a latent variable that we compute and we do not have measurements. However, we can computed the likelihood of E.

$$\begin{aligned} P(E|S \cap D) &= \alpha P(E)P(S|E)P(D|E) \qquad &&= \alpha P(E)L(E|S \cap D) \\ L(E|S \cap D) &= (S|E)P(D|E) \end{aligned}$$

   (b) Then we look at the root note C. Given $F = f_5$:

$$P(C|E \cap F) = \alpha P(C)P(E|C)P(F|C)$$

$$P(e|c_k) = \sum_{i=1}^{3} P(e_i|c_k)L(e_i)$$

$$P'(c_k) = P(c_k|e \cap f_5) = \alpha P(c_k)\left(\sum_{i=1}^{3} P(e_i|c_k)L(e_i)\right)P(f_5|c_k)$$

   (c) We see the the evidence for C comes from
      i. Evidence coming from E and its sub-tree
      ii. Evidence from everywhere else.

$$\begin{aligned} P_E(C) &= \alpha P(C)P(F|C) \\ \boldsymbol{P}(\boldsymbol{E}) &= \boldsymbol{P}(\boldsymbol{E}|\boldsymbol{C})\boldsymbol{P}_{\boldsymbol{E}}(\boldsymbol{C}) \end{aligned}$$

   (d) Suppose if we have the instantiations $S = s_3$ and $D = d_2$

$$P'(e_i) = \alpha P(e_i)P(s_3|e_i)P(d_2|e_i)$$

3 Evidence & Message Passing

1. New concepts to deal with complex networks with intermediate nodes:
   - **Evidence** is the information that we have at a node -this may be gathered through instantiation (exact value or virtual evidence), or inferred from passing messages. Evidence is unnormalized probabilities so the absolute values are meaningless, but they are useful for making comparisons.
   - **Messages** is the information (evidence) passed between nodes to provide evidence to another node.
2. **Theorem:** Let $(\mathbb{G}, P)$ be a Bayesian network whose DAG is a tree, where $\mathbb{G} = (V, E)$, and a be a set of values of a subset $A \subset V$.
   (a) λ **messages**: For each child Y of X, $\forall x \in X$
   $$\lambda_Y(x) = \sum_y P(y|x)\lambda(y)$$
   (b) λ **values**:
      i. If $X \in A$ and X is instantiated to $\hat{x}$
      $$\lambda(\hat{x}) = 1, \text{ for } x = \hat{x}$$
      $$\lambda(x) = 0, \text{ for } x \neq \hat{x}$$
      ii. if $X \notin A$ and X is a leaf, $\forall x \in X$,
      $$\lambda(x) = 1$$
      iii. If $X \notin A$ and X is not a leaf, $\forall x \in X$
      $$\lambda(x) = \prod_{U \in CH_X} \lambda_U(x),$$
      where CH_X denotes the set of the children of X.
   (c) π **messages**: If Z is the parent of X, then $\forall z \in Z$
   $$\pi_X(z) = \pi(z) \prod_{U \in CH_Z - \{X\}} \lambda_U(z)$$
   (d) π **values**:
      i. If $X \in A$ and X is instantiated to $\hat{x}$:
      $$\pi(\hat{x}) = 1, \text{ for } x = \hat{x}$$
      $$\pi(x) = 0, \text{ for } x \neq \hat{x}$$
      ii. If $X \notin A$ and X is the root, $\forall x \in X$
      $$\pi(X) = P(x)$$

iii. If $X \notin A$, X is not the root and Z is the parent of X, $\forall x \; in X$

$$\pi(x) = \sum_z P(x|z)\pi_X(z)$$

(e) Given the definitions, for each variable X, we have for all values of x,

$$P(x|a) = \alpha\lambda(x)\pi(x)$$

3. The π values are basically evidence from the parents and it generalises the concept of prior. The λ values are basically evidence from the descendents and it generalises the concept of likelihood probability.

4. Mnemonic: **p**i (π), **p**rior, and "**p**arent" all start with letter "p"; and **l**ambda (λ), **l**ikelihood, and "**l**ad" all start with "l". Further, prior comes *before* so from parents.

5. If we use virtual evidence at the leaf nodes, we can use the conditioning equation (above b(iii))

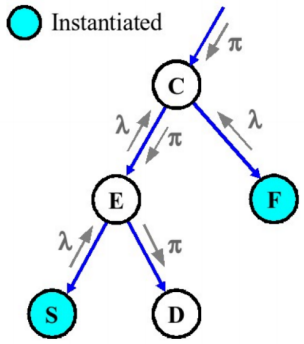


Figure 2: Upon instantiation of a node, we can propagate the λ and π messages

6. Important equations:

$$\boldsymbol{\lambda_S(E) = \lambda(S)P(S|E)} \qquad \boldsymbol{\pi(E) = P(E|C)\pi_E(C)}$$

$$\lambda(e_i) = \prod_{CH_E}\left[\sum_j \lambda(s_j)P(s_j|e_i)\right] \qquad \pi(e_i) = \sum_j\left[P(e_i|c_j)\pi(c_j)\prod_{k\backslash E}\lambda_k(c_j)\right]$$

7. Notation

   (a) $\boldsymbol{\lambda_F(c)}$ means λ evidence from node F to node C.

   (b) $\boldsymbol{\pi_F(c)}$ means π from all nodes besides F for node C. (This can alternatively viewed as the information from C to F)

3.1 Probability Propagation in Trees

3.1.1 Operative Formulas

1. If B is a child of A, B has k possible values, and A has m possible values, then for $1 \leq j \leq m$, the λ message from B to A is given by

$$\lambda_B(a_j) = \sum_{i=1}^{k} P(b_i|a_j)\lambda(b_i)$$

2. If B is a child of A and A has m possible values, then for $1 \leq j \; leqm$ the π message from A to B is given by

$$\pi_B(a_j) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated, but not for } a_j \\ \frac{P'(a_j)}{\lambda_B(a_j)} & \text{if } A \text{ is not instantiated} \end{cases}$$

where $P'(a_j)$ is the current conditional probability of a_j based on the variables thus fr instantiated.

3. If B is a variable with k possible values, $s(B)$ is the set of B's children, then for $1 \leq i \leq k$ the λ value of B is given by

$$\lambda(b_i) = \begin{cases} \prod_{C \in s(B)} \lambda_C(b_i) & \text{if } B \text{ is not instantiated} \\ 1 & \text{if } B \text{ is instantiated for } b_i \\ 0 & \text{if } B \text{ is instantiated, but not for } b_i \end{cases}$$

4. If B is a variable with k possible values, A is the parent of B, and A has m possible values, the for $1 \leq i \leq k$ the π value of B is given by

$$\pi(b_i) = \sum_{j=1}^{m} P(b_i|a_j)\pi_B(a_j)$$

5. If B is a variable with k possible values, then for $1 \leq i \leq k$, $P'(b_i)$, the conditional probability of b_i based on the variables thus far instantiated, is given by

$$P'(b_i) = \alpha\lambda(b_i)\pi(b_i)$$

3.1.2 Initialization

1. Set all λ messages and λ values to 1.

2. If the root A has m possible values, then for $1 \leq j \leq m$, set

$$\pi(a_j) = P(a_j)$$

3. For all children B of the root A, post a new π message to B using operative formula (2).

3.1.3 Updating

When a variable is instantiated or a λ or π message is received by a variable, one of the following updating procedures is used:

(a) **Instantiation.** If a variable B is instantiated for b_j, then
   i. Set $P'(b_j) = 1$ and for $i \neq j$, set $P'(b_i) = 0$
   ii. Compute $\lambda(B)$ using operative formula (3).
   iii. Post a new λ message to B's parents using operative formula (1).
   iv. Post a new π message to B's children using operative formula (2).

(b) **Upward Propagation.** If a variable B receives a new λ message from one of its children and if B is not already instantiated, then
   i. Compute the new value for $\lambda(B)$ using operative formula (3)
   ii. Compute the new value of $P'(B)$ using operative formula (5)
   iii. Post new λ message to B's parents using operative formula (1)
   iv. Post new π messages to B's other children using operative formula (2)

(c) **Downward Propagation.** If a variable B receives a new π message from its parent and if B is not already instantiated, then
   i. Compute the new value of $\pi(B)$ using operative formula (4)
   ii. Compute the new value of $P'(B)$ using operative formula (5)
   iii. Post new π messages to B's children using operative formulat (2)

4 Single Connected Networks

1. A DAG is singly connected if there is at most one path between any two nodes. In a singly-connected network, each node can have more than 1 parents.

2. The main properties of these networks are:

   (a) The parents of a node are always independent given their common child (i.e. they don't have a common parent). This lets us calculate their joint probability as the product of their marginals.

   (b) When updating evidence of a node, the belief propagated through the net to update all nodes is guaranteed to reach a steady state.

3. An example of a link matrix for a node with multiple parents looks as follows:

$$\boldsymbol{P(e|w,c)} = \begin{bmatrix} P(e_1|w_1,c_1) & P(e_1|w_1,c_2) & P(e_1|w_2,c_1) & P(e_1|w_2,c_2) \\ (e_2|w_1,c_1) & P(e_2|w_1,c_2) & P(e_2|w_2,c_1) & P(e_2|w_2,c_2) \\ P(e_3|w_1,c_1) & P(e_3|w_1,c_2) & P(e_3|w_2,c_1) & P(e_3|w_2,c_2) \end{bmatrix}$$

4. To calculate the π evidence of a node with 2 parents (assuming independence between the parents):

$$\pi(\boldsymbol{E}) = \boldsymbol{P(e|w,c)\pi_e(w,c)} = \boldsymbol{P(e|w,c)\pi_e(w)\pi_e(c)}$$

5. To calculate the λ evidence of a node with 2 parents, we have to calculate one λ message for each of the parents. If c has parents a and b, then the evidence from c to a is as follows:

$$\lambda_c(a_i) = \sum_{j=1}^{n} \pi_c(b_j) \sum_{k=1}^{m} P(c_k|a_i,b)\lambda(c_k)$$

   where n is the number of values that b takes, and m is the number of values c takes.

6. **Theorem:** Let $(\mathbb{G}, P)$ be a Bayesian network that is singly-connected, where $\mathbb{G} = (V, E)$, and a be a set of values of a subset $A \subset V$.

   (a) λ **messages**: For each child Y of X, $\forall x \in X$

$$\lambda_Y(x) = \sum_{y} \left[\sum_{w_1,\ldots,w_k} \left(P(y|x,w_i,\ldots,w_k) \prod_{i=1}^{k} \pi_Y(w_i) \right) \right] \lambda(y)$$

   where $\{W_i\}_{i=1}^{k}$ are the other parents of Y.

   (b) λ **values**:

      i. If $X \in A$ and X is instantiated to $\hat{x}$

$$\lambda(\hat{x}) = 1, \text{ for } x = \hat{x}$$
$$\lambda(x) = 0, \text{ for } x \neq \hat{x}$$

ii. if $X \notin A$ and X is a leaf, $\forall x \in X$,

$$\lambda(x) = 1$$

iii. If $X \notin A$ and X is not a leaf, $\forall x \in X$

$$\lambda(x) = \prod_{U \in CH_X} \lambda_U(x),$$

where CH_X denotes the set of the children of X.

(c) π **messages**: If Z is the parent of X, then $\forall z \in Z$

$$\pi_X(z) = \pi(z) \prod_{U \in CH_Z - \{X\}} \lambda_U(z)$$

(d) π **values**:

i. If $X \in A$ and X is instantiated to $\hat{x}$:

$$\pi(\hat{x}) = 1, \text{ for } x = \hat{x}$$
$$\pi(x) = 0, \text{ for } x \neq \hat{x}$$

ii. If $X \notin A$ and X is the root, $\forall x \in X$

$$\pi(X) = P(x)$$

iii. If $X \notin A$, X is not the root and $\{Z_i\}_{i=1}^{j}$ are the parents of X, $\forall x \; in X$

$$\pi(x) = \sum_{z_1,\ldots,z_j} \left(P(x|z_1,\ldots,z_j) \prod_{i=1}^{j} \pi_X(z_i) \right)$$

(e) Given the definitions, for each variable X, we have for all values of x,

$$P(x|a) = \alpha \lambda(x) \pi(x)$$

7. **Blocked Path (very important to understand this):**

- For a diverging path: when the node is instantiated, it will block the passing message between other nodes.
- For a converging path: it is blocked when there is no λ evidence on the child node, but unblocked when there is λ evidence or when the node is instantiated.

$$\lambda_C(a_i) = \sum_{j=1}^{m} \pi_C(b_j) \sum_{k=1}^{n} P(c_k|a_i \cap b_j) \lambda(c_k)$$

$$\lambda(c_k) = 1, \forall c_k \Rightarrow \lambda_C(a_i) = \sum_{j=1}^{m} \pi_C(b_j)$$

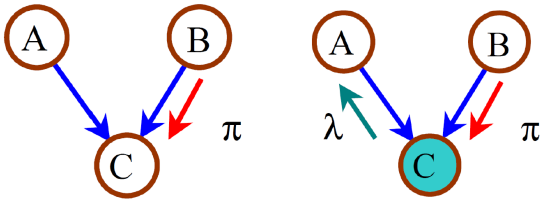


Figure 3: Converging Connections

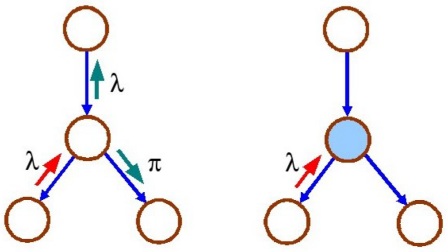


Figure 4: Diverging Connection

4.1 Probability Propagation

4.1.1 The Operating Equations for Probability Propagation

1. **The λ Message.** If C is a child of A and B, then the λ message from C to A is given by

$$\lambda_C(a_i) = \sum_{j=1}^{m} \pi_C(b_j) \sum_{k=1}^{n} P(c_k|a_i, b_j)\lambda(c_k)$$

$$\boldsymbol{\lambda_C(A) = \lambda(C)P(C|A)}$$

$$\lambda_C(a_i) = \sum_{j=1}^{m} \pi_C(b_j)\lambda_C(a_i \cap b_j)$$

2. The π Message: If C is a child of A, the π message from A to C is:

$$\pi_C(a_i) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_i \\ 0 & \text{if } A \text{ is instantiated but not for } a_i \\ P'(a_i)/\lambda_C(a_i) & \text{if } A \text{ is not instantiated} \end{cases}$$

where $P'(a_i)$ is defined to be the current conditional probability of a_i based on the variables thus far instantiated.

3. The λ Evidence: If C is a node with n children $D_1, \ldots, D_n$, then the λ evidence for C is

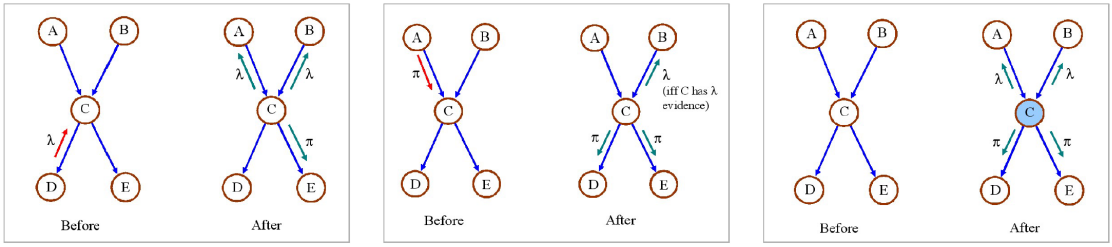
$$\lambda(c_k) = \begin{cases} 1 & \text{if } C \text{ is instantiated for } c_k \\ 0 & \text{if } C \text{ is instantiated but not for } c_k \\ \prod_i \lambda_{D_i}(c_k) & \text{if } C \text{ is not instantiated} \end{cases}$$

4. The π Evidence: If C is a child of two parents A and B, the pi evidence for C is:

$$\pi(c_k) = \sum_{i=1}^{l} \sum_{j=1}^{m} P(c_k|a_i, b_j)\pi_C(a_i)\pi_C(b_i)$$
$$\boldsymbol{\pi(C) = P(C|A)\pi_C(A)}$$

5. The Posterior Probabilities: If C is a variable, the posterior probability of C based on the evidence received is

$$P'(c_k) = \alpha\lambda(c_k)\pi(c_k)$$



(a) Upward Propagation (b) Downward propagation (c) Instantiation

Figure 5: Probability propagation

4.1.2 Initialization

1. All λ messages, π messages and λ values are set to 1.

2. For all root nodes A, the π values are set to the prior probabilities.

$$\pi(a_j) = P(a_j)$$

3. Post and propagate the π messages from the root nodes using downward propagation with operative formula 2.

4.1.3 Updating

1. **Upward Propagation.** A node C receives λ from a child and C is not instantiated:
   - Compute the new $\lambda(C)$ using operative equation (3).
   - Compute the new posterior probability $P'(C)$ using operative equation (5).
   - Post a λ message to all C's parents.
   - Post a π message to all C's other children.
2. **Downward Propagation.** If a variable C receives a π message from one parent:
   - If C is not instantiated:
     - Compute a new value for array $\pi(C)$ using operative equation (4)
     - Compute a new value for $P(C)$ using operative equation (5)
     - Post a π message to each child.
   - If there is λ evidence in C
     - Post a λ message to the other parents.
3. **Instantiation.** If C is instantiated for state c_k,
   - Set $P'(c_j) = 0, \forall j \neq k$
   - Set $P'(c_k) = 1$
   - Compute $\lambda(C)$ using operative equation (3)
   - Post λ and π message to each parent and each child of C respectively.

5 Building Networks from Data

1. Building networks from data:

   (a) **Expert knowledge**: We can obtained the network structure would be known or readily obtainable from an expert. This approach is highly subjective and reliant on the expert advice.

   (b) **Spanning tree algorithms**: Main idea is to start with a set of nodes to which we add arcs until a complete network is created.

      i. The nodes that are joined by arcs are depedent and those not are at least conditionally independent. Hence the strategy is to add arcs that connect variables that are most dependent.

      ii. For every pair of variables calculate the dependency. Then join the nodes in dependency order, provided the resulting structure has no loops

      iii. The algorithm minimises the KL divergence between the network joint probability and the data joint probability

2. **Adding Causal Directions:**

   (a) If a node is considered a root node, we will point all arrows from it.

   (b) On the whole, cause is a semantic entity that needs human intervention to determine.

3. **Measures of dependencies**

   Weighted measures are used to reduce dependencies for less likely values.

   - $L1$ **dependency measure**:

$$\begin{aligned} \textit{unweighted:} \qquad & Dep(A,B) = \sum_{A\times B} \left| P(a_i \cap b_j) - P(a_i)P(b_j) \right| \\ \textit{weighted:} \qquad & Dep(A,B) = \sum_{A\times B} P(a_i \cap b_j) \times \left| P(a_i \cap b_j) - P(a_i)P(b_j) \right| \end{aligned}$$

   - $L2$ **dependency measure**:

$$\begin{aligned} \textit{unweighted:} \qquad & Dep(A,B) = \sum_{A\times B} \Big(P(a_i \cap b_j) - P(a_i)P(b_j) \Big)^2 \\ \textit{weighted:} \qquad & Dep(A,B) = \sum_{A\times B} P(a_i \cap b_j) \times \Big(P(a_i \cap b_j) - P(a_i)P(b_j) \Big)^2 \end{aligned}$$

   As the probabilities becomes small they contribute less to the dependency, and this effect is acceptable since we have little information on rare events. The weighted version of the $L1$ and $L2$ further reduces the dependency for low probability values.

- **Kullback-Leibler Measure (mutual entropy)**
  - It is zero when two variables are completely independent
  - It is positive and increasing with dependency when applied to probability distributions
  - It is independent of the actual value of probability
  - Can be computed as follows

$$Dep(A,B) = \sum_{A\times B} P(a_i \cap b_j) \log_2 \left[\frac{P(a_i \cap b_j)}{P(a_i)P(b_j)} \right]$$

- **Correlation**
  - Measures only linear dependency whereas mutual entropy can characterise higher order dependencies more accurately.
  - Can be computed as follows:

$$C(A,B) = \frac{\Sigma_{AB}}{\sqrt{\sigma_A \sigma_B}}$$

$$\Sigma_{AB} = \frac{1}{N-1} \sum_{i=1}^{N} (a_i - \bar{a}_i)(b_i - \bar{b}_i)$$

$$\sigma_A = \frac{1}{N-1} \sum_{i=1}^{N} (a_i - \bar{a}_i)^2$$

6 Cause and Independence

1. Independence: In a network, if there is no path between two variables, they are independent.

    - Two variables can be connected by a path in a network and still be independent as the conditional probability can express no dependency.
    - In theory, the absence of an arc is more significant than the presence of an arc. However, in theory we avoid arcs with very low dependency in networks.
    - The spanning tree algorithm can be terminated when the dependecy becomes too low to be significant.

2. (d-separation) Let $\mathbb{G} = (V, E)$ be a DAG, $A \subseteq V$ and X and Y be distinct nodes in $V - A$. X and Y are d-separated by A in $\mathbb{G}$ if every chain between X and Y is blocked by A.

3. Possible configurations of connected triplets:

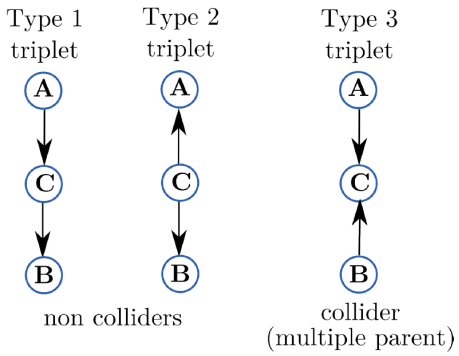


Figure 6: Possible configurations of connected triplets.

(a) **Non-colliders**: If C is instantiated, no messages pass from A to B.

(b) **Colliders:** The nodes A and B are independent if there is no information on C.

- Marginal Independence: We can measure the dependence between A and B using all the data. If this is low, the configuration may be multiple parent.
- Conditional Independence: We can partition data according to states of C and compute the dependency between A and B. If this is high, the configuration may be multiple parent.

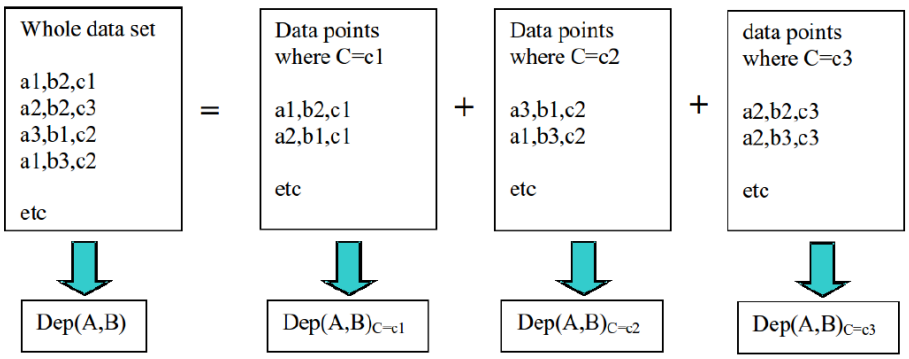


Figure 7: Practical computation.

4. **Determining Causal Directions:**

   (a) Compute the maximally weighted spanning tree

   (b) For each connected triplet in the spanning tree

   - Compute the joint probability of the triplet
   - Compute the marginal dependence and condition dependence
   - If marginal dependence is low and the conditional dependence is high, put in causal directions corresponding to a collider

   (c) Propagate the causal arrows as far as possible

5. Structure and Parameter Learning

   - Bayesian networks
     - Combined both structure and parameter learning.
     - We can express our knowledge by choosing the structure or we can learn it through data.
     - Incorporate known causal relations but have the potential to learn cause from data.
   - Neural networks
     - Offers only parameter learning.
     - It is difficult to embed knowledge or infer structure from a neural network.
     - Most applicable when we have no causal knowledge but we cannot extract causal information from them.
   - Rule based system (logic based)
     - Have just structure but they are not scalable and difficult to optimise.
     - Most applicable when we have causal knowledge as these systems can represent it well

7 Model Accuracy

1. The model may be evaluated using **maximum likelihood estimation**.

$$\begin{aligned} P(Ds|Bn) &= \prod_{data} P(Bn) \\ &= \prod_{i} P(x_i)P(y_i|x_i) \text{ (case of two variables)} \end{aligned}$$

   - Above is the general formulation, but to avoid underflow errors, one may take the log likelihood. If the log we take is base 2, then the log likelihood is known as the information measure since $\log_2$ N bits are required to represent integers up to N.

$$\log(P(Ds|Bn)) = \sum_{data} log(P(Bn))$$

   - Ds is a given dataset, Bn is a our BayesNet model, and $\prod_{data} P(Bn)$ is the joint probability of the model multiplied over all values in the dataset.
   - Larger models get higher scores, so size becomes a confounding variable.

2. **Minimum Description Length Score** (MDLScore, also known as Bayesian Information Criterion) penalises larger models to provide a fair comparison for smaller models. The size of the model is characterized by the number of parameters.

$$MDLScore = \frac{|B_n|}{2} \log_2 N - \log_2 P(D_s|B_n)$$

   - A gentle reminder of principle of parsimony or Occam's Razor: A simple model is preferred to a complex model.
   - The first part of the above equation is the average number of bits required to store the values.
   - **Example**. If a network has a parent and a child, and we have 4 datapoints ($N = 4$, each node takes 2 values), the parent has a prior probability of 2 parameters, and the conditional table has $2 \times 2 = 4$. Therefore, $Size(Bn|Ds) = |Bn|log_2(4)/2 = [(2-1)+(2-1)\times 2]log_2(4)/2 = 3 \times 2/2 = 3$.
   - MDL Score is not an absolute measure. It can only be used to compare two models of the same data set.
   - **need to add why this is a better measure than euclidean distance (from tutorial)**

3. **Measuring Predictive Accuracy**: Split the data into training data and test data, then use the test data to measure the network accuracy. It can be used to minimize the number of variables in a spanning tree.

- Build a spanning tree and obtain an ordering of the nodes starting at the root.
- Remove all the arcs.
- Add arcs in the order of their dependency.
- If the predictive accuracy is good enough, stop. Else, go back to 3rd step.

8 Approximate Inference

1. For highly dependent data, we can:
   (a) Model all dependencies - Propagation of probabilities is difficult or infeasible.
   (b) Maximally weighted spanning tree - Does not model dependencies accurately but message passing terminates in one pass and it is very fast.
2. Problems with loops in networks:
   (a) Looping messages.
   (b) Independence of multiple parents.

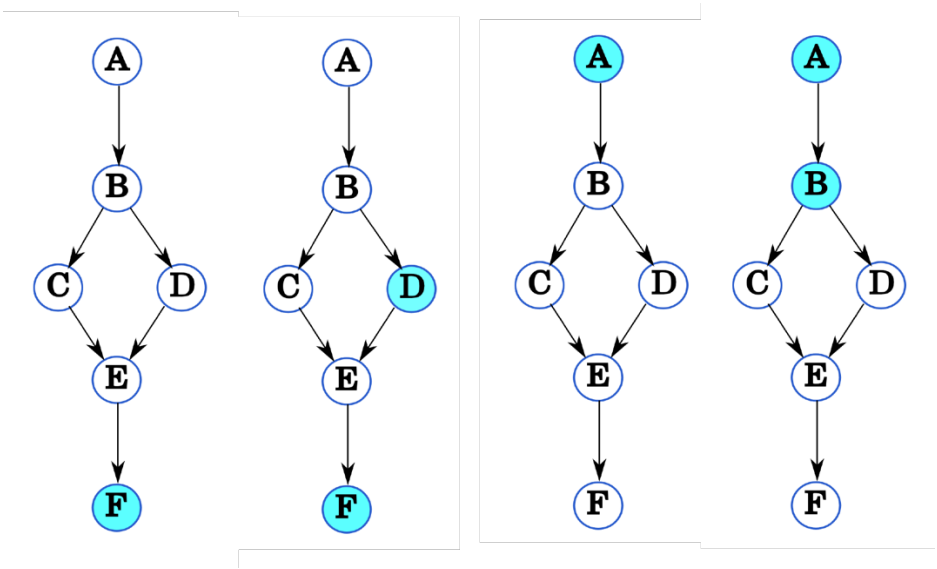


Figure 8: (Left half) When F is instantiated, the messages will not stop propagating. However, if one of B, C or D is instantiated, exact propagation is possible. (Right half) When only A is instantiated, C and D are no independent and the pi evidence at E and F is not correct. Exact propagation can be still carried out if B,C, and D are instantiated.

3. Approximate Inference Methods:
   - Node deletion (Seletive Bayes Network)
   - Loopy belief propagation
   - Hidden / latent node placement
4. **Node Deletion and Selective Bayes Network**: Main idea is to use a subset of the variables.
   - Start with a network with all variables then deleting any variables and testing the improvement.

- Alternatively, add variables incrementally and test the performance for each network.

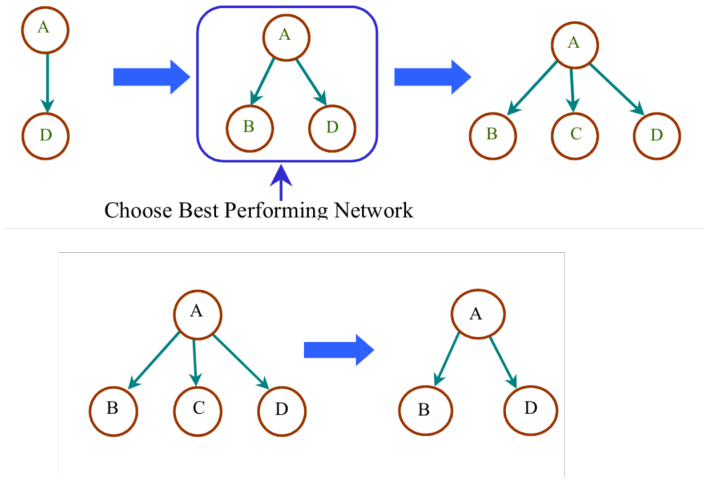


Figure 9: (Top Row) Adding incrementally. (Bottom row) Deletion incrementally.

5. **Loopy Belief Propagation**: Include all arcs expressing significant dependency and allow propagation to continue until

   - The probability distributions reach a stable state; or
   - A limiting number of iterations has occurred (i.e. there may be no steady state)

6. **Hidden Nodes**: If any two children of a parent are not conditionally independent, they can be separated by hidden node.

   - Switch nodes: Hidden nodes can act as switches to simplify the networks.
   - A network can always perform as well with a hidden node as it can without.

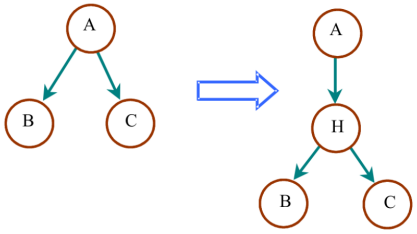


Figure 10: Hidden Nodes to introduce conditional independence.

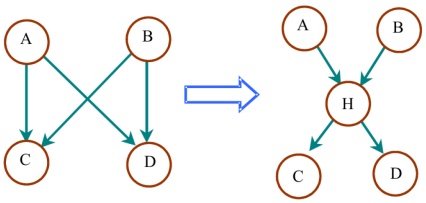


Figure 11: Hidden Nodes to untangle double loops and acts as switch nodes.

8.1 Using Hidden Nodes

1. To create a hidden note, we need to decide:
   - How many states it should have.
   - Identify values for the 3 new linked matrices introduced.
2. Determination of number of states:
   - Number of states for hidden node will be comparable to the number of states of the nodes it is separating.
   - Expect the hidden node to have at least the same number of states of its parent.
   - Linked matrices with too many states will have low probabilities for some states. Hence, start with large number of states and then reduce them.
3. Calculating the linked matrices (conditional probabilities)
   (a) Given the estimates of $P(H|A)$, $P(B|H)$, $P(C|H)$ and a set of data points $[a_i, b_j, c_k]$.
   (b) Use each b_j, c_k to compute $P'(A)$ from the network, calculate and accumulate the error

$$E = [P'(A) - P(a_i)]^2$$

   (c) Minimize E over the data set by adjusted $P(H|A)$, $P(B|H)$, $P(C|H)$ using the gradient descent algorithm

$$P(h_k|a_j) \rightarrow P(h_k|a_j) - \mu \frac{\partial E}{\partial P(h_k|a_j)}$$
$$P(b_j|h_k) \rightarrow P(b_j|h_k) - \mu \frac{\partial E}{\partial P(b_j|h_k)}$$
$$P(c_j|h_k) \rightarrow P(c_j|h_k) - \mu \frac{\partial E}{\partial P(c_j|h_k)}$$

   (d) Using gradient descent may not actually arrive at the optimal answer due to the following issues:

- Distributions will no longer sum to 1.
- Individual probability values may be greater than 1 or less than 0.
- Conditional probability matrices need to be normalized and this may compromise finding an optimum solution.

(e) The propagation strategies for calculating errors as per Figure 12. If strategies are alternated during optimization, it produces annealing behaviour

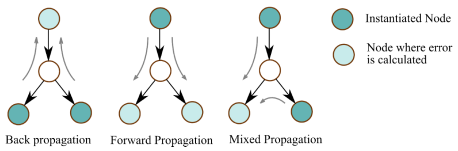


Figure 12: Propagation Strategies in Calculating Errors.

4. Hidden nodes can be used to remove loops. We can always reduce any network to a singly connected form by this method as in Figure 13, but the performance will be increasingly dependent on the training data and training process.

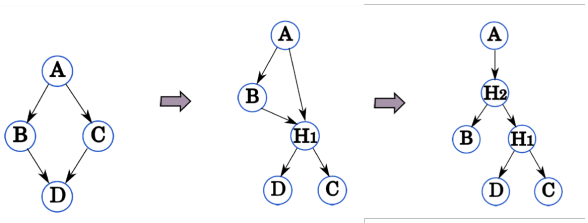


Figure 13: Propagation Strategies in Calculating Errors.

5. Limitations for hidden nodes
   - We may need too many hidden nodes to model all dependencies through hidden nodes.
   - The number of states in the hidden node will become large.
6. Criteria for introducing hidden nodes
   - We measure the conditional dependency of each pair of children given the parents.
   - If the dependency is high, we expect benefits from introduction of the hidden nodes. Else, we can ignore the dependency.

9 Exact Inference

There are methods to do approximate inference

1. **Cutset conditioning**
    - **Method**: (1) Identify a cut set of nodes, which if instantiated makes propoagation terminate correctly, (2) If the node cannot be instantiated (no data), the network is broken into several networks, one for each possible states of the node
    - **Problems**: (1) Computation time expands exponentially; (2) Not enough data to do it sufficiently.
2. **Node clustering**
    - **Method**: Combining the nodes into a new variable
    - **Problems**: The increase number of states in the new node limits the applicability of this method as the number of nodes goes up exponentially.
3. **Join trees**:
    - **Method**: Basically a generalization of the node clustering method. However, it is to find a systematic way to join the variables such that the probability propagation is possible.

10 Join Trees

10.1 Preliminaries

1. **Potential Representations.** Let V be a finite set of propositional variables and P be a joint probability distribution of V. Suppose W_i, $1 \leq i \leq p$ is a collection of subsets of V, and ψ is a function that assigns a unique real number to every combination of values of the propositional variables in W_i. Then we have

$$P(V) = K \prod_{i=1}^{p} \psi(W_i)$$

2. **The running intersection property.** Let V be a set and $\{W_i$ such that $1 \leq i \leq p$ be an **ordered** set of subsets of V. Then we have

$$S_i = W_i \cap (W_1 \cup W_2 \cup \cdots \cup W_{i-1})$$
$$R_i = W_i - S_i$$

3. **Conditional probabilities**

   - Let V be a finite set of propositional variables, P is a joint probability distribution of V and $\{W_i \forall i \leq\leq p\}$ an ordered set of subsets of V

$$P(R_i|S_i) = P(W_i|S_i)$$

   - Let V be a finite set of propositional variables, P is a joint probability distribution of V and $(\{W_i \forall i \leq\leq p\}, \psi)$ a potential representation of P

$$P(R_p|S_p) = \frac{\psi(W_p)}{\sum_{R_p} \psi(W_p)}$$

4. **Potential and running intersection property.** Let V be a finite set of propositional variables and P be a joint probability distribution of V and $(W_i \forall 1 \leq i \leq p, \psi)$ a potential representation of P. If the ordering $[W_1, \ldots, W_p]$ has the running intersection property

$$P(V) = P(W_1) \prod_{i=2}^{p} P(R_i|S_i)$$

10.2 Join Tree Construction from a DAG

A ordered subset of the variables can be found from a causal graph as fllow

1. Moralisation

   - For each child node with multiple parents, we need to ensure that the child and the parents are in the same clique.

- Hence we need to marry the parents by constructing an edge joining the parents.

2. Triangulation
   - A chordal/triangulated graph is one in which all cycles of four or more nodes have a chord, i.e. an edge that is not part of the cycle but connects two nodes of the cycle.
   - This is to ensure that the marginal distribution for a variable appearing in multiple nodes are consistent and equal.
3. Identify the cliques of the resulting graph
   - A clique is a maximal set of nodes in which every node is connected to every other.
4. Find an ordering with the running intersection property. Can be done using a search algorithm
   (a) Start with a clique containing a root of the original network
   (b) Find all nodes that can be its children - i.e. those for which any variable appearing higher up in the evolving tree is in the parent
   (c) Recursively search from the children till all cliques are joined
5. For each clique, Clq, find the set of its variables $\{X_i\}$ whose parents $Pa(X_i)$ also belongs to the clique and initialize the clique potential function as follows:

$$\psi(Clq) = \psi(W_i) = \prod_{\{X_i\}} P(X|Pa(X))$$

10.3 Probability propagation in join trees

1.
2.
3. Computing $P(R|S)$
   (a) We can calculate the value with the following:

$$P(R|S) = \frac{\psi(W_i)}{\sum_R \psi(W_i)}$$

   (b) The clique potential function represent all accumulated evidence for that node. The formula above is a normalization that creates the link matrix from the accumulated evidence.
4. Sending λ message