Bayesuvius,
a small visual dictionary of Bayesian Networks

Robert R. Tucci
www.ar-tiste.xyz

November 26, 2020

Figure 1: View of Mount Vesuvius from Pompeii

Figure 2: Mount Vesuvius and Bay of Naples

Contents

0.1 Foreword

Welcome to Bayesuvius! a proto-book uploaded to github.

A different Bayesian network is discussed in each chapter. Each chapter title is the name of a B net. Chapter titles are in alphabetical order.

This is a volcano in its early stages. First version uploaded to a github repo called Bayesuvius on June 24, 2020. First version only covers 2 B nets (Linear Regression and GAN). I will add more chapters periodically. Remember, this is a moonlighting effort so I can't do it all at once.

For any questions about notation, please go to Notational Conventions section.

Requests and advice are welcomed.

Thanks for reading this.
Robert R. Tucci
www.ar-tiste.xyz

0.2 Definition of a Bayesian Network

A **directed graph** $G = (V, E)$ consists of two sets, V and E. V contains the **vertices (nodes)** and E contains the **edges (arrows)**. An arrow $a \rightarrow b$ is an ordered pair (a, b) where $a, b \in V$.

The **parents** of a node x are those nodes a such that there are arrows $a \rightarrow x$. The **children** of a node x are those nodes b such that there are arrows $x \rightarrow b$. A **root node** is a node with no parents. A **leaf node** is a node with no children. The **neighbors** of a node x is the set of parents and children of x.

A **path** is a set of nodes that are connected by arrows, so that all nodes have 1 or 2 neighbors, but only two nodes (**open path**) or zero nodes (**closed path**) have only one neighbor. A **directed path** is a path in which all the arrows point in the same direction. A **loop** is a closed path;i.e., a path in which all nodes have exactly 2 neighbors. A **cycle** is a directed loop. A **Directed Acyclic Graph (DAG)** is a directed graph that has no cycles.

A **fully connected directed graph** is a directed graph in which every node has all other nodes as neighbors. Figs.3 and 4 show 2 different ways of drawing the same directed graph, a fully connected graph with 4 nodes. Note that a convenient way to label the nodes of a fully connected directed graph with N nodes is to point arrows from $\underline{x}_k$ to $\underline{x}_j$ where $j = 0, 1, 2, \ldots, N-1$ and $k = j-1, j-2, \ldots, 0$.

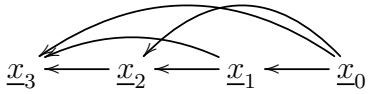


Figure 3: Fully connected directed graph with 4 nodes, drawn as a line.

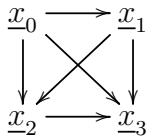


Figure 4: Fully connected directed graph with 4 nodes, drawn as a square.

A **connected graph** is a graph for which there is no way of separating the nodes into two sets so that there is no arrow from one set to the other. A **tree** is a directed graph in which all nodes have a single parent except for a single node called the "root" node which has no parents. A **polytree** is a DAG with no loops.

A **Bayesian network (bnet)** consists of a DAG and a **Transition Probability Matrix (TPM)** associated with each node of the graph. A TPM is often called a **Conditional Probability Table (CPT)**. The **structure** of a bnet is its DAG alone, sans the TPMs. The **skeleton** of a bnet is the undirected graph beneath the bnet's DAG.

In this book, random variables are indicated by underlined letters and their values by non-underlined letters. Each node of a bnet is labelled by a random variable. Thus, $\underline{x} = x$ means that node $\underline{x}$ is in state x.

Some sets of nodes associated with each node $\underline{a}$ of a bnet

- $ch(\underline{a})$ = children of $\underline{a}$.
- $pa(\underline{a})$ = parents of $\underline{a}$.
- $nb(\underline{a}) = pa(\underline{a}) \cup ch(\underline{a})$ = neighbors of $\underline{a}$.
- $de(\underline{a}) = \cup_{n=1}^{\infty} ch^n(\underline{a}) = ch(\underline{a}) \cup ch \circ ch(\underline{a}) \cup \ldots$, descendants of $\underline{a}$.
- $an(\underline{a}) = \cup_{n=1}^{\infty} pa^n(\underline{a}) = pa(\underline{a}) \cup pa \circ pa(\underline{a}) \cup \ldots$, ancestors of $\underline{a}$.

In this book, we will use $\underline{a}.$ to indicate a **multi-node (node set, node array)** $\underline{a}. = (\underline{a}_j)_{j=0,1,\ldots,na-1}$. We will often treat multinodes as if they were sets, and combine them with the usual set operators. For instance, for two multinodes $\underline{a}.$ and $\underline{b}.$, we define $\underline{a}. \cup \underline{b}.$, $\underline{a}. \cap \underline{b}.$, $\underline{a}. - \underline{b}.$ and $\underline{a}. \subset \underline{b}.$ in the obvious way. We will indicate a singleton set (single node multi-node) $\underline{a}. = \{\underline{a}\}$ simply by $\underline{a}. = \underline{a}$. For instance, $\underline{a}. - \underline{b} = \underline{a}. - \{\underline{b}\}$.

The TPM of a node $\underline{x}$ of a bnet is a matrix of probabilities $P(\underline{x} = x | pa(\underline{x}) = a.)$.

A bnet with nodes $\underline{x}.$ represents a probability distribution

$$P(x.) = \prod_j P(\underline{x}_j = x_j | (\underline{x}_k = x_k)_{k:\underline{x}_k \in pa(\underline{x}_j)}) \ . \tag{1}$$

Note that for a fully connected bnet with N nodes, Eq.(1) becomes

$$P(x.) = \prod_{j=0}^{N-1} P(x_j | (x_k)_{k=j-1,j-2,\ldots,0}) \ . \tag{2}$$

For example, if $N = 4$, Eq.(2) becomes

$$P(x_0, x_1, x_2, x_3) = P(x_3|x_2, x_1, x_0) P(x_2|x_1, x_0) P(x_1|x_0) P(x_0) \ . \tag{3}$$

We see that Eq.(2) is just the chain rule for conditional probabilities.

Given an arbitrarily large dataset of samples for the random variables $(\underline{x}_i)_{i=0,1,\ldots,N-1}$, there may be several bnets that fit the data well, but only one is used by Nature. That single one is called a **causal (or causally correct) Bayesian Network**. In this book, whenever we speak of causal issues, we will assume, often without mentioning it, that the correct bnet is being used.

0.3 Notational Conventions and Preliminaries

Some abbreviations frequently used throughout this book.

- bnet= B net= Bayesian Network
- CPT = Conditional Probabilities Table, same as TPM
- DAG = Directed Acyclic Graph
- i.i.d.= independent identically distributed.
- TPM= Transition Probability Matrix, same as CPT

Define $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ to be the integers, real numbers and complex numbers, respectively.

For $a < b$, define $I_{\mathbb{Z}}$ to be the integers in the interval I, where $I = [a,b], [a,b), (a,b], (a,b)$ (i.e, I can be closed or open on either side).

$A_{>0} = \{k \in A : k > 0\}$ for $A = \mathbb{Z}, \mathbb{R}$.

Random variables will be indicated by underlined letters and their values by non-underlined letters. Each node of a bnet will be labelled by a random variable. Thus, $\underline{x} = x$ means that node $\underline{x}$ is in state x.

It is more conventional to use an upper case letter to indicate a random variable and a lower case letter for its state. Thus, $X = x$ means that random variable X is in state x. However, we have opted in this book to avoid that notation, because we often want to define certain lower case letters to be random variables or, conversely, define certain upper case letters to be non-random variables.

$P_{\underline{x}}(x) = P(\underline{x} = x) = P(x)$ is the probability that random variable $\underline{x}$ equals $x \in S_{\underline{x}}$. $S_{\underline{x}}$ is the set of states (i.e., values) that $\underline{x}$ can assume and $n_{\underline{x}} = |S_{\underline{x}}|$ is the size (aka cardinality) of that set. Hence,

$$\sum_{x \in S_{\underline{x}}} P_{\underline{x}}(x) = 1 \tag{4}$$

$$P_{\underline{x},\underline{y}}(x,y) = P(\underline{x} = x, \underline{y} = y) = P(x,y) \tag{5}$$

$$P_{\underline{x}|\underline{y}}(x|y) = P(\underline{x} = x|\underline{y} = y) = P(x|y) = \frac{P(x,y)}{P(y)} \tag{6}$$

Kronecker delta function: For x, y in discrete set S,

$$\delta(x,y) = \begin{cases} 1 \text{ if } x = y \\ 0 \text{ if } x \neq y \end{cases} \tag{7}$$

Dirac delta function: For $x, y \in \mathbb{R}$,

$$\int_{-\infty}^{+\infty} dx\ \delta(x-y) f(x) = f(y) \tag{8}$$

The TPM of a node of a bnet can be either a discrete or a continuous probability distribution. To go from continuous to discrete, one replaces integrals over states of a node by sums over new states, and Dirac delta functions by Kronecker delta functions. More precisely, consider a function $f : [a, b] \to \mathbb{R}$. Express $[a, b]$ as a union of small, disjoint (except for one point) closed sub-intervals (bins) of length Δx. Name one point in each bin to be the representative of that bin, and let $S_{\underline{x}}$ be the set of all the bin representatives. This is called discretization or binning. Then

$$\frac{1}{(b-a)} \int_{[a,b]} dx\ f(x) \to \frac{\Delta x}{(b-a)} \sum_{x \in S_{\underline{x}}} f(x) = \frac{1}{n_{\underline{x}}} \sum_{x \in S_{\underline{x}}} f(x)\ . \tag{9}$$

Both sides of last equation are 1 when $f(x) = 1$. Furthermore, if $y \in S_{\underline{x}}$, then

$$\int_{[a,b]} dx\ \delta(x-y) f(x) = f(y) \to \sum_{x \in S_{\underline{x}}} \delta(x,y) f(x) = f(y)\ . \tag{10}$$

Indicator function (aka Truth function):

$$\mathbb{1}(\mathcal{S}) = \begin{cases} 1 \text{ if } \mathcal{S} \text{ is true} \\ 0 \text{ if } \mathcal{S} \text{ is false} \end{cases} \tag{11}$$

For example, $\delta(x, y) = \mathbb{1}(x = y)$.

$$\vec{x} = (x[0], x[1], x[2] \ldots, x[nsam(\vec{x}) - 1]) = x[:] \tag{12}$$

$nsam(\vec{x})$ is the number of samples of $\vec{x}$. $\underline{x}[\sigma] \in S_{\underline{x}}$ are i.i.d. (independent identically distributed) samples with

$$x[\sigma] \sim P_{\underline{x}} \ \text{(i.e. } P_{\underline{x}[\sigma]} = P_{\underline{x}}) \tag{13}$$

$$P(\underline{x} = x) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x[\sigma] = x) \tag{14}$$

Hence, for any $f : S_{\underline{x}} \to \mathbb{R}$,

$$\sum_{x} P(\underline{x} = x) f(x) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} f(x[\sigma]) \tag{15}$$

If we use two sampled variables, say $\vec{x}$ and $\vec{y}$, in a given bnet, their number of samples $nsam(\vec{x})$ and $nsam(\vec{y})$ need not be equal.

$$P(\vec{x}) = \prod_{\sigma} P(x[\sigma]) \tag{16}$$

$$\sum_{\vec{x}} = \prod_{\sigma} \sum_{x[\sigma]} \tag{17}$$

$$\partial_{\vec{x}} = [\partial_{x[0]}, \partial_{x[1]}, \partial_{x[2]}, \ldots, \partial_{x[nsam(\vec{x})-1]}] \tag{18}$$

$$P(\vec{x}) \approx [\prod_x P(x)^{P(x)}]^{nsam(\vec{x})} \tag{19}$$

$$= e^{nsam(\vec{x})\sum_x P(x)\ln P(x)} \tag{20}$$

$$= e^{-nsam(\vec{x})H(P_{\underline{x}})} \tag{21}$$

$$f^{[1,\partial_x,\partial_y]}(x,y) = [f, \partial_x f, \partial_y f] \tag{22}$$

$$f^{+} = f^{[1,\partial_x,\partial_y]} \tag{23}$$

For probabilty distributions $p(x), q(x)$ of $x \in S_{\underline{x}}$

- Entropy:

$$H(p) = -\sum_x p(x)\ln p(x) \geq 0 \tag{24}$$

- Kullback-Liebler divergence:

$$D_{KL}(p \parallel q) = \sum_x p(x)\ln\frac{p(x)}{q(x)} \geq 0 \tag{25}$$

- Cross entropy:

$$CE(p \to q) = -\sum_x p(x)\ln q(x) \tag{26}$$

$$= H(p) + D_{KL}(p \parallel q) \tag{27}$$

Normal Distribution: $x, \mu, \sigma \in \mathbb{R}$, $\sigma > 0$

$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{28}$$

Uniform Distribution: $a < b$, $x \in [a,b]$

$$\mathcal{U}(x;a,b) = \frac{1}{b-a} \tag{29}$$

Expected Value and Variance

Given a random variable $\underline{x}$ with states $S_{\underline{x}}$ and a function $f : S_{\underline{x}} \to \mathbb{R}$, define

$$E_{\underline{x}}[f(\underline{x})] = E_{x\sim P(x)}[f(x)] = \sum_x P(x)f(x) \tag{30}$$

$$Var_{\underline{x}}[f(\underline{x})] = E_{\underline{x}}\left[(f(\underline{x}) - E_{\underline{x}}[f(\underline{x})])^2\right] \tag{31}$$

$$= E_{\underline{x}}[f(\underline{x})^2] - (E_{\underline{x}}[f(\underline{x})])^2 \tag{32}$$

$$E[\underline{x}] = E_{\underline{x}}[\underline{x}] \tag{33}$$

$$Var[\underline{x}] = Var_{\underline{x}}[\underline{x}] \tag{34}$$

Conditional Expected Value

Given a random variable $\underline{x}$ with states $S_{\underline{x}}$, a random variable $\underline{y}$ with states $S_{\underline{y}}$, and a function $f : S_{\underline{x}} \times S_{\underline{y}} \to \mathbb{R}$, define

$$E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})] = \sum_x P(x|\underline{y}) f(x, \underline{y}) \ , \tag{35}$$

$$E_{\underline{x}|\underline{y}=y}[f(\underline{x}, y)] = E_{\underline{x}|y}[f(\underline{x}, y)] = \sum_x P(x|y) f(x, y) \ . \tag{36}$$

Note that

$$\begin{aligned} E_{\underline{y}}[E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})]] &= \sum_{x,y} P(x|y) P(y) f(x, y) && (37) \\ &= \sum_{x,y} P(x, y) f(x, y) && (38) \\ &= E_{\underline{x},\underline{y}}[f(\underline{x}, \underline{y})] \ . && (39) \end{aligned}$$

Law of Total Variance

Claim 1 *Suppose $P : S_{\underline{x}} \times S_{\underline{y}} \to [0, 1]$ is a probability distribution. Suppose $f : S_{\underline{x}} \times S_{\underline{y}} \to \mathbb{R}$ and $f = f(x, y)$. Then*

$$Var_{\underline{x},\underline{y}}(f) = E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(f)] + Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[f]) \ . \tag{40}$$

In particular,

$$Var_{\underline{x}}(x) = E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(x)] + Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[x]) \ . \tag{41}$$

proof:

Let

$$A = \sum_y P(y) \left(\sum_x P(x|y) f \right)^2 \ . \tag{42}$$

Then

$$\begin{aligned} Var_{\underline{x},\underline{y}}(f) &= \sum_{x,y} P(x, y) f^2 - \left(\sum_{x,y} P(x, y) f \right)^2 && (43) \\ &= \begin{cases} \sum_{x,y} P(x, y) f^2 - A \\ + \left(A - \left(\sum_{x,y} P(x, y) f \right)^2 \right) \end{cases} && (44) \end{aligned}$$

$$E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(f)] = \sum_y P(y)\left(\sum_x P(x|y)f^2 - \left(\sum_x P(x|y)f\right)^2\right) \quad (45)$$

$$= \sum_{x,y} P(x,y)f^2 - A \quad (46)$$

$$Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[f]) = \sum_y P(y)\left(\sum_x P(x|y)f\right)^2 - \left(\sum_y P(y)\sum_x P(x|y)f\right)^2 \quad (47)$$

$$= A - \left(\sum_{x,y} P(x,y)f\right)^2 \quad (48)$$

QED

$\langle \underline{x}, \underline{y} \rangle$ notation, for covariances of any two random variables $\underline{x}, \underline{y}$.

Mean value of $\underline{x}$

$$\langle \underline{x} \rangle = E_{\underline{x}}[\underline{x}] \quad (49)$$

Signed distance of $\underline{x}$ to its mean value

$$\Delta\underline{x} = \underline{x} - \langle \underline{x} \rangle \quad (50)$$

Covariance of $(\underline{x}, \underline{y})$

$$\langle \underline{x}, \underline{y} \rangle = \langle \Delta\underline{x}\Delta\underline{y} \rangle = Cov(\underline{x}, \underline{y}) \quad (51)$$

Variance of $\underline{x}$

$$Var(\underline{x}) = \langle \underline{x}, \underline{x} \rangle \quad (52)$$

Standard deviation or $\underline{x}$

$$\sigma_{\underline{x}} = \sqrt{\langle \underline{x}, \underline{x} \rangle} \quad (53)$$

Correlation of $(\underline{x}, \underline{y})$

$$\rho_{\underline{x},\underline{y}} = \frac{\langle \underline{x}, \underline{y} \rangle}{\sqrt{\langle \underline{x}, \underline{x} \rangle \langle \underline{y}, \underline{y} \rangle}} \quad (54)$$

---

linear regression

$\underline{y}$ = true value
$\underline{\hat{y}}$ = estimator
$\underline{\epsilon}$ = residual

$$\underline{\hat{y}} = \beta_0 + \sum_{j=1}^{n} \beta_j \underline{x}_j \quad (55)$$

$$\underline{y} = \underline{\hat{y}} + \underline{\epsilon} \quad (56)$$

Assume

$$\langle \underline{x}_j, \underline{\epsilon} \rangle = 0 \tag{57}$$

for all j.

For $k = 1, \ldots, n$,

$$\langle \underline{x}_k, \underline{y} \rangle = \sum_{j=1}^{n} \beta_j \langle \underline{x}_k, \underline{x}_j \rangle \ . \tag{58}$$

Let $\underline{x}^n$ and β^n be column vectors. Then

$$\langle \underline{x}^n, \underline{y} \rangle = \langle \underline{x}^n, (\underline{x}^n)^T \rangle \beta^n \ , \tag{59}$$

$$\beta^n = \langle \underline{x}^n, (\underline{x}^n)^T \rangle^{-1} \langle \underline{x}^n, \underline{y} \rangle \ . \tag{60}$$

Sigmoid function

For $x \in \mathbb{R}$,

$$\mathrm{sig}(x) = \frac{1}{1 + e^{-x}} \tag{61}$$

$\mathcal{N}(!a)$ will denote a normalization constant that does not depend on a. For example, $P(x) = \mathcal{N}(!x)e^{-x}$ where $\int_0^\infty dx \ P(x) = 1$.

A **one hot** vector of zeros and ones is a vector with all entries zero with the exception of a single entry which is one. A **one cold** vector has all entries equal to one with the exception of a single entry which is zero. For example, if $x^n = (x_0, x_1, \ldots, x_{n-1})$ and $x_i = \delta(i, 0)$ then x^n is one hot.

Short Summary of Boolean Algebra.

See Ref.[35] for more info about this topic.

Suppose $x, y, z \in \{0, 1\}$. Define

$$x \text{ or } y = x \vee y = x + y - xy \ , \tag{62}$$

$$x \text{ and } y = x \wedge y = xy \ , \tag{63}$$

and

$$\text{not } x = \overline{x} = 1 - x \ , \tag{64}$$

where we are using normal addition and multiplication on the right hand sides.[1]

Actually, since $x \wedge y = xy$, we can omit writing the symbol $\wedge$. The symbol $\wedge$ is useful to exhibit the symmetry of the identities, and to remark about the analogous identities for sets, where $\wedge$ becomes intersection $\cap$ and $\vee$ becomes union $\cup$. However, for practical calculations, $\wedge$ is an unnecessary nuisance.

Since $x \in \{0, 1\}$,

$$P(\overline{x}) = 1 - P(x) \ . \tag{65}$$

[1] Note the difference between $\vee$ and modulus 2 addition $\oplus$. For $\oplus$ (aka XOR): $x \oplus y = x + y - 2xy$.

Associativity	$x \vee (y \vee z) = (x \vee y) \vee z$ $x \wedge (y \wedge z) = (x \wedge y) \wedge z$
Commutativity	$x \vee y = y \vee x$ $x \wedge y = y \wedge x$
Distributivity	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
Identity	$x \vee 0 = x$ $x \wedge 1 = x$
Annihilator	$x \wedge 0 = 0$ $x \vee 1 = 1$
Idempotence	$x \vee x = x$ $x \wedge x = x$
Absorption	$x \wedge (x \vee y) = x$ $x \vee (x \wedge y) = x$
Complementation	$x \wedge \overline{x} = 0$ $x \vee \overline{x} = 1$
Double negation	$\overline{(\overline{x})} = x$
De Morgan Laws	$\overline{x} \wedge \overline{y} = \overline{(x \vee y)}$ $\overline{x} \vee \overline{y} = \overline{(x \wedge y)}$

Table 1: Boolean Algebra Identities

Clearly, from analyzing the simple event space $(x, y) \in \{0, 1\}^2$,

$$P(x \vee y) = P(x) + P(y) - P(x \wedge y) \; . \tag{66}$$

Definition of various entropies used in Shannon Information Theory

- **(plain) Entropy of $\underline{x}$**

$$H(\underline{x}) = -\sum_x P(x) \ln P(x) \tag{67}$$

This quantity measures the spread of $P_{\underline{x}}$.

- **Conditional Entropy of $\underline{y}$ given $\underline{x}$**

$$\begin{aligned} H(\underline{y}|\underline{x}) &= -\sum_{x,y} P(x,y) \ln P(y|x) & (68) \\ &= H(\underline{y}, \underline{x}) - H(\underline{x}) & (69) \end{aligned}$$

This quantity measures the conditional spread of $\underline{y}$ given $\underline{x}$.

- **Mutual Information (MI) of $\underline{x}$ and $\underline{y}$**

$$H(\underline{y} : \underline{x}) = \sum_{x,y} P(x,y) \ln \frac{P(x,y)}{P(x)P(y)} \tag{70}$$

$$= H(\underline{x}) + H(\underline{y}) - H(\underline{y}, \underline{x}) \tag{71}$$

This quantity measures the correlation between $\underline{x}$ and $\underline{y}$.

- **Conditional Mutual Information (CMI)[2] of $\underline{x}$ and $\underline{y}$ given $\underline{\lambda}$**

$$H(\underline{y} : \underline{x}|\underline{\lambda}) = \sum_{x,y,\lambda} P(x,y,\lambda) \ln \frac{P(x,y|\lambda)}{P(x|\lambda)P(y|\lambda)} \tag{72}$$

$$= H(\underline{x}|\underline{\lambda}) + H(\underline{y}|\underline{\lambda}) - H(\underline{y}, \underline{x}|\underline{\lambda}) \tag{73}$$

This quantity measures the conditional correlation of $\underline{x}$ and $\underline{y}$ given $\underline{\lambda}$.

- **Kullback-Liebler Divergence from $P_{\underline{x}}$ to $P\underline{y}$.**

Assume random variables $\underline{x}$ and $\underline{y}$ have the same set of states $S_{\underline{x}} = S_{\underline{y}}$. Then

$$D_{KL}(P_{\underline{x}} \,||\, P_{\underline{y}}) = \sum_{x} P_{\underline{x}}(x) \ln \frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} \tag{74}$$

This measures a non-symmetric distance between the probability distributions $P_{\underline{x}}$ and $P_{\underline{y}}$. $D_{KL}(P_{\underline{x}} \,||\, P_{\underline{y}})$ is non-negative and equals zero iff $P_{\underline{x}} = P_{\underline{y}}$.

[2] CMI can be read as "see me".

0.4 Navigating the ocean of Judea Pearl's Books

Many of the greatest ideas in the bnet field were invented by Judea Pearl and his collaborators. Thus, this book is heavily indebted to those great scientists. Those ideas have had no clearer and more generous expositor than Judea Pearl himself.

Pearl has written 4 books that I have used in writing Bayesuvius. His 1988 book Ref.[18] dates back to his pre-causal period. That book I used to learn about topics such as d-separation, belief propagation, Markov-blankets, and noisy-ORs. 3 other books that he wrote later, in his causal period, are:

1. In 2000 (1st ed.), and 2013 (2nd ed.), Pearl published what is so far his most technical and exhaustive book on the subject of causality, Ref.[19].

2. In 2016, he released together with Glymour and Jewell, a less advanced "primer" on causality, Ref.[21].

3. In 2018, he released together with Mackenzie his lovely "The Book of Why", Ref.[22].

Those 3 books I used to learn about causality topics such as do-calculus, backdoor and front-door adjustments, linear deterministic bnets with exogenous noise, and counterfactuals.

Chapter 1

ARACNE-structure learning

This chapter is based on Ref.[11].

The ARACNE algo is an algo for learning the structure of a bnet from data. The algo considers data samples for n random variables $(x_i)_{i=0,1,...,n-1}$, and estimates the mutual information $MI_{i,j} = H(\underline{x}_i : \underline{x}_j)$ between every pair of nodes. The set UG is initialized to contain all the edges of a fully connected undirected graph. Next the algo removes from UG every edge with $MI_{i,j} < \epsilon$ for some threshold $0 < \epsilon << 1$. Then the algo examines every triplet of edges in UG, and marks for removal the edge of the triplet with the smallest MI. Finally, the algo removes from UG all edges marked for removal. Each triplet is analyzed irrespective of whether its edges have been marked for removal when considering a prior triplet. Thus the network constructed by the algorithm is independent of the order in which the triplets are examined. Some of the unoriented edges in UG can be given an orientation using the same techniques used to orient edges in constraint based structure learning (see Chapter 42).

Ref.[11] incorrectly claims that removing the smaller of 3 MI's is "an application" of the Data Processing Inequality (DPI) of Shannon Information Theory. See Chapter 26 for more info about DPI. Note that DPI is only valid for a Markov chain, and not all triplets of random variables are in a Markov chain. Removing the smaller of 3 numbers does not require DPI.

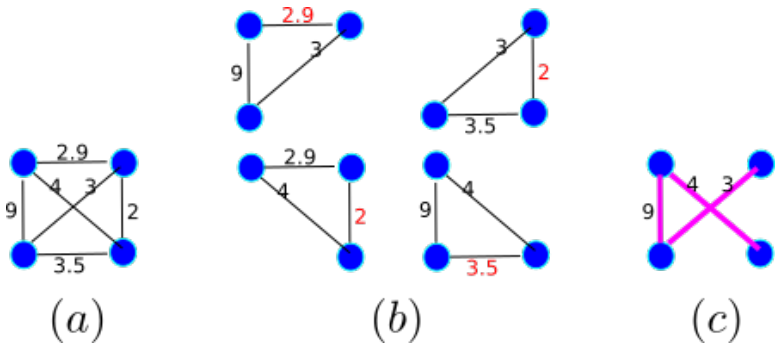


Figure 1.1: An example where the ARACNE algo gives a Chow-Liu tree. (a) Fully connected undirected graph with weights $MI_{i,j}$ along the edges. (b) All 4 possible triplets of edges with nonzero weights. Edges marked for removal have their weights printed in red.(c) Final structure.

Fig.1.1 gives an example of the application of the ARACNE algo.

See Chapter 7 on Chow-Liu trees (CLT). A CLT is just a maximum spanning tree where the weights are mutual informations $MI_{i,j}$ estimated from data.

Sometimes, the outcome of the ARACNE algo is a CLT. For example, Fig.1.1 (a) was considered in Chapter 7 on CLTs, and the CLT algo also gave Fig.1.1 (c) as the final structure.

According to Ref.[11], the ARACNE algo sometimes yields a polytree (i.e., a connected graph with no loops). It may even yield a structure with loops. Hence, it does not always yield a CLT.

By breaking all cliques (i.e., fully connected subgraphs) with 3 edges and 3 nodes, ARACNE breaks all cliques with 3 or more nodes. However, cliques are not uncommon in Nature, especially 3 node cliques. Cliques become less likely in Nature the bigger the number of nodes they have *after* 3. Therefore, a nice generalization of ARACNE would be to list all 4 node cliques, and break each of them by eliminating their edge with the smallest MI. This will have the effect of breaking all cliques with 4 or more nodes but keeping 3 node cliques. One could also break some, not all, of the 3 node cliques, by consecutively removing the clique-breaking-edge with the smallest MI of all edges of all 3 node cliques. Let β stand for banned clique number of nodes. Then the current ARACNE has $\beta = 3$. We are suggesting that a β of 4 might be more likely to occur in Nature.

Chapter 2

Backdoor Adjustment

The backdoor (BD) adjustment theorem is proven in Chapter 11 from the rules of do-calculus. The goal of this chapter is to give examples of the use of that theorem. We will restate the theorem in this chapter, sans proof. There is no need to understand the theorem's proof in order to use it. However, you will need to skim Chapter 11 in order to familiarize yourself with the notation used to state the theorem. This chapter also assumes that you are comfortable with the rules for checking for d-separation. Those rules are covered in Chapter 12.

Suppose we have access to data that allows us to estimate a probability distribution $P(x., y., z.)$. Hence, the variables $\underline{x}., \underline{y}., \underline{z}.$ are all observed (i.e, not hidden). Then we say that the backdoor $\underline{z}.$ satisfies the **backdoor adjustment criterion** relative to $(\underline{x}., \underline{y}.)$ if

1. All paths from $\underline{x}.$ to $\underline{y}.$ that start with an arrow pointing into $\underline{x}.$, are blocked by $\underline{z}.$.

2. $\underline{z}. \cap de(\underline{x}.) = \varnothing$.

Claim 2 Backdoor Adjustment Theorem

If $\underline{z}.$ satisfies the backdoor criterion relative to $(\underline{x}., \underline{y}.)$, then

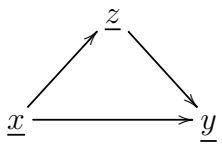
$$P(y.|\rho\underline{x}. = x.) = \sum_{z.} P(y.|x., z.)P(z.) \tag{2.1}$$

$$= \sum_{z.} \left\{ \begin{array}{c} \\ \underline{z}. = z. \\ \\ \\ \underline{x}. = x. \longrightarrow \underline{y}. \\ \\ \end{array} \right\} \tag{2.2}$$

proof: See Chapter 11
QED

Examples:

1.

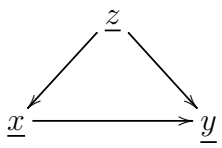


(2.3)

BD criterion satisfied if $\underline{x}. = \underline{x}, \underline{y}. = \underline{y}, \underline{z}. = \emptyset$. No adjustment necessary.

$$P(y|\rho\underline{x} = x) = P(y|x) \tag{2.4}$$

2.

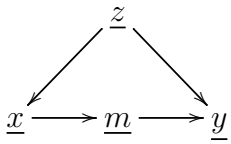


(2.5)

BD criterion satisfied if $\underline{x}. = \underline{x}, \underline{y}. = \underline{y}, \underline{z}. = \underline{z}$.

Note that here the backdoor formula adjusts the parents of $\underline{x}$..

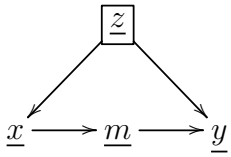
3.



(2.6)

BD criterion satisfied if $\underline{x}. = \underline{x}, \underline{y}. = \underline{y}, \underline{z}. = \underline{z}$.

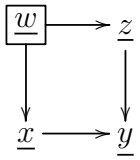
4.



(2.7)

BD criterion is impossible to satisfy if $\underline{x}. = \underline{x}, \underline{y}. = \underline{y}$. However, the front-door criterion can be satisfied. See Chapter 15.

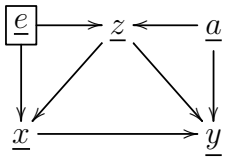
5.



(2.8)

BD criterion satisfied if $\underline{x}. = \underline{x}, \underline{y}. = \underline{y}, \underline{z}. = \underline{z}$. Note that here the backdoor formula cannot adjust the single parent $\underline{w}$ of $\underline{x}$ because it is hidden, but we are able to block the backdoor path by conditioning on $\underline{z}$ instead.

6.

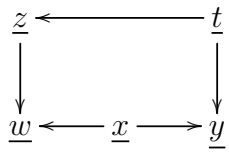


(2.9)

Conditioning on $\underline{z}$ blocks backdoor path $\underline{x}-\underline{z}-\underline{y}$, but opens path $\underline{x}-\underline{e}-\underline{z}-\underline{a}-\underline{y}$ because $\underline{z}$ is a collider for that path. That path is blocked if we also condition on $\underline{a}$, which is possible because $\underline{a}$ is observed. In conclusion, the BD criterion is satisfied if $\underline{x}. = \underline{x}$, $\underline{y}. = \underline{y}$ and $\underline{z}. = (\underline{z}, \underline{a})$.

Conditioning on the parents of $\underline{x}.$ is often enough to block all backdoor paths. However, sometimes some of the parents are unobserved and one must condition on other nodes that are not parents of $\underline{x}.$ in order to satisfy the BD criterion.

7.



(2.10)

No need to control anything because only possible backdoor path is blocked by collider $\underline{w}$. Hence,

$$P(y|\rho\underline{x} = x) = P(y|x) \ . \tag{2.11}$$

However, if for some reason we want to control $\underline{t}$, we can do so. We can't control $\underline{w}$ though, because $\underline{w} \in de(\underline{x})$. Thus, the BD criterion is satisfied if $\underline{x}. = \underline{x}$, $\underline{y}. = \underline{y}$ and $\underline{z}. = \underline{t}$. Therefore,

$$P(y|\rho\underline{x} = x) = \sum_t P(y|x,t)P(t) \ . \tag{2.12}$$

---

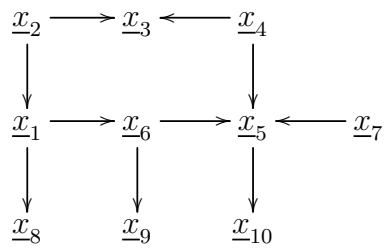
8. Discuss reasons why multiple possible sets $\underline{z}.$ that satisfy the BD criterion can be advantageous.

   - Can evaluate $P(y.|\rho\underline{x}. = x.)$ multiple ways and compare the results. This is a test that the causal bnet is correct.
   - It might be easier or less expensive to get data for some $\underline{z}.$ more than for others.

---

9. (Taken from online course notes Ref.[6])

Consider the bnet



(2.13)

If $\underline{x}. = \underline{x}_1$ and $\underline{y}. = \underline{x}_5$, find all possible adjustment multinodes $\underline{z}.$ that satisfy the BD criterion. Ans:

- $\emptyset$
- $\underline{x}_2$
- $\underline{x}_4$
- $\underline{x}_2, \underline{x}_4$
- $\underline{x}_2, \underline{x}_3$
- $\underline{x}_3, \underline{x}_4$
- $\underline{x}_2, \underline{x}_3, \underline{x}_4$

Add $\underline{x}_7$ to each of the previous 7 possible $\underline{z}.$. This gives a total of 14 possible adjustment multinodes $\underline{z}.$.

Chapter 3

Back Propagation (Automatic Differentiation)

General Theory

Jacobians

Suppose $f : \mathbb{R}^{nx} \to \mathbb{R}^{nf}$ and

$$y = f(x) \ . \tag{3.1}$$

Then the Jacobian $\frac{\partial y}{\partial x}$ is defined as the matrix with entries[1]

$$\left[\frac{\partial y}{\partial x}\right]_{i,j} = \frac{\partial y_i}{\partial x_j} \ . \tag{3.2}$$

Jacobian of function composition. Supoose $f : \mathbb{R}^{nx} \to \mathbb{R}^{nf}$, $g : \mathbb{R}^{nf} \to \mathbb{R}^{ng}$. If

$$y = g \circ f(x) \ , \tag{3.3}$$

then

$$\frac{\partial y}{\partial x} = \frac{\partial g}{\partial f}\frac{\partial f}{\partial x} \ . \tag{3.4}$$

Right hand side of last equation is a product of two matrices so order of matrices is important.

Let

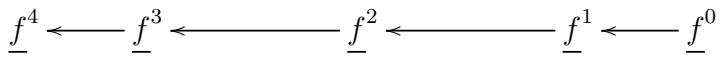
$$y = f^4 \circ f^3 \circ f^2 \circ f^1(x) \ . \tag{3.5}$$

This function composition chain can be represented by the bnet Fig.3.1(a) with TPMs

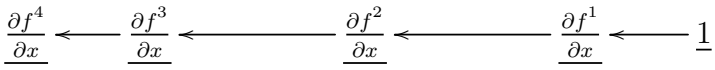
$$P(f^\mu | f^{\mu-1}) = \mathbb{1}(f^\mu = f^\mu(f^{\mu-1})) \tag{3.6}$$

for $\mu = 1, 2, 3, 4$.

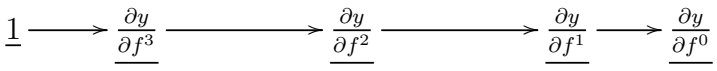
[1] Mnemonic for remembering order of indices: i in numerator/j in denominator becomes index i/j of Jacobian matrix.



(a) Composition



(b) Forward-p



(c) Back-p

Figure 3.1: bnets for function composition, forward propagation and back propagation for $nf = 5$ nodes.

Note that

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial f^3}\frac{\partial f^3}{\partial f^2}\left[\frac{\partial f^2}{\partial f^1}\frac{\partial f^1}{\partial x}\right] \tag{3.7}$$

$$= \frac{\partial y}{\partial f^3}\left[\frac{\partial f^3}{\partial f^2}\frac{\partial f^2}{\partial x}\right] \tag{3.8}$$

$$= \left[\frac{\partial y}{\partial f^3}\frac{\partial f^3}{\partial x}\right] \tag{3.9}$$

$$= \frac{\partial y}{\partial x} . \tag{3.10}$$

This forward propagation can be represented by the bnet Fig.3.1(b) with node TPMs

$$P(\frac{\partial f^{\mu+1}}{\partial x} \mid \frac{\partial f^{\mu}}{\partial x}) = \mathbb{1}(\frac{\partial f^{\mu+1}}{\partial x} = \frac{\partial f^{\mu+1}}{\partial f^{\mu}}\frac{\partial f^{\mu}}{\partial x}) \tag{3.11}$$

for $\mu = 1, 2, 3$.

Note that

$$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial f^3}\frac{\partial f^3}{\partial f^2}\right]\frac{\partial f^2}{\partial f^1}\frac{\partial f^1}{\partial x} \tag{3.12}$$

$$= \left[\frac{\partial y}{\partial f^2}\frac{\partial f^2}{\partial f^1}\right]\frac{\partial f^1}{\partial x} \tag{3.13}$$

$$= \left[\frac{\partial y}{\partial f^1}\frac{\partial f^1}{\partial x}\right] \tag{3.14}$$

$$= \frac{\partial y}{\partial x}\ . \tag{3.15}$$

This back propagation can be represented by the bnet Fig.3.1(c) with node TPMs

$$P(\frac{\partial y}{\partial f^\mu} \mid \frac{\partial y}{\partial f^{\mu+1}}) = \mathbb{1}(\frac{\partial y}{\partial f^\mu} = \frac{\partial y}{\partial f^{\mu+1}}\frac{\partial f^{\mu+1}}{\partial f^\mu}) \tag{3.16}$$

for $\mu = 2, 1, 0$.

$\frac{\partial f^{\mu+1}}{\partial f^\mu}$ is a Jacobian matrix so the order of multiplication matters. In forward prop, it pre-multiplies, and in back prop it post-multiplies.

Application to Neural Networks

Absorbing b_i^λ into $w_{i|j}$.

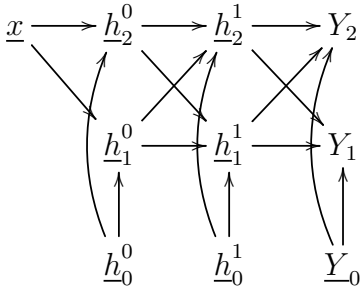


Figure 3.2: Nodes $\underline{h}_0^0, \underline{h}_0^1, \underline{Y}_0$ are all set to 1. They allow us to absorb b_i^λ into the first column of $w_{i|j}^\lambda$.

Below are, printed in blue, the TPMs for the nodes of a NN bnet, as given in Chapter 31. For all hidden layers $\lambda = 0, 1, \ldots, \Lambda - 2$,

$$P(h_i^\lambda \mid h^{\lambda-1}) = \delta\left(h_i^\lambda, \mathcal{A}_i^\lambda(\sum_j w_{i|j}^\lambda h_j^{\lambda-1} + b_i^\lambda)\right) \tag{3.17}$$

for $i = 0, 1, \ldots, nh(\lambda) - 1$. For the output visible layer $\lambda = \Lambda - 1$:

$$P(Y_i \mid h_{\cdot}^{\Lambda-2}) = \delta\left(Y_i, \mathcal{A}_i^{\Lambda-1}(\sum_j w_{i|j}^{\Lambda-1} h_j^{\Lambda-2} + b_i^{\Lambda-1})\right) \tag{3.18}$$

for $i = 0, 1, \ldots, ny - 1$.

For each λ, replace the matrix $w_{\cdot|\cdot}^{\lambda}$ by the augmented matrix $[b^{\lambda}_{\cdot}, w_{\cdot|\cdot}^{\lambda}]$ so that the new $w_{\cdot|\cdot}^{\lambda}$ satisfies

$$w_{i|0}^{\lambda} = b_i^{\lambda} \tag{3.19}$$

Let the nodes $\underline{h}_0^{\lambda}$ for all λ and $\underline{Y}_0$ be root nodes (so no arrows pointing into them). For each λ, draw arrows from $\underline{h}_0^{\lambda}$ to all other nodes in that same layer. Draw arrows from $\underline{Y}_0$ to all other nodes in that same layer.

After performing the above steps, the TPMs, printed in blue, for the nodes of the NN bnet are as follows:

For all hidden layers $\lambda = 0, 1, \ldots, \Lambda - 2$,

$$P(h_0^{\lambda}) = \delta(h_0^{\lambda}, 1) \ , \tag{3.20}$$

and

$$P(h_i^{\lambda} \mid h_{\cdot}^{\lambda-1}, h_0^{\lambda} = 1) = \delta\left(h_i^{\lambda}, \mathcal{A}_i^{\lambda}(\sum_j w_{i|j}^{\lambda} h_j^{\lambda-1})\right) \tag{3.21}$$

for $i = 1, \ldots, nh(\lambda) - 1$. For the output visible layer $\lambda = \Lambda - 1$:

$$P(Y_0) = \delta(Y_0, 1) \ , \tag{3.22}$$

and

$$P(Y_i \mid h_{\cdot}^{\Lambda-2}, Y_0 = 1) = \delta\left(Y_i, \mathcal{A}_i^{\Lambda-1}(\sum_j w_{i|j}^{\Lambda-1} h_j^{\Lambda-2})\right) \tag{3.23}$$

for $i = 1, 2, \ldots, ny - 1$.

From here on, we will rename y above by $Y = \hat{y}$ and consider samples $y[i]$ for $i = 0, 1, \ldots, nsam - 1$. The Error (aka loss or cost function) is

$$\mathcal{E} = \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} |Y_i - y_i[\sigma]|^2 \tag{3.24}$$

To perform simple gradient descent, one uses:

$$(w_{i|j}^{\lambda})' = w_{i|j}^{\lambda} - \eta \frac{\partial \mathcal{E}}{\partial w_{i|j}^{\lambda}} \ . \tag{3.25}$$

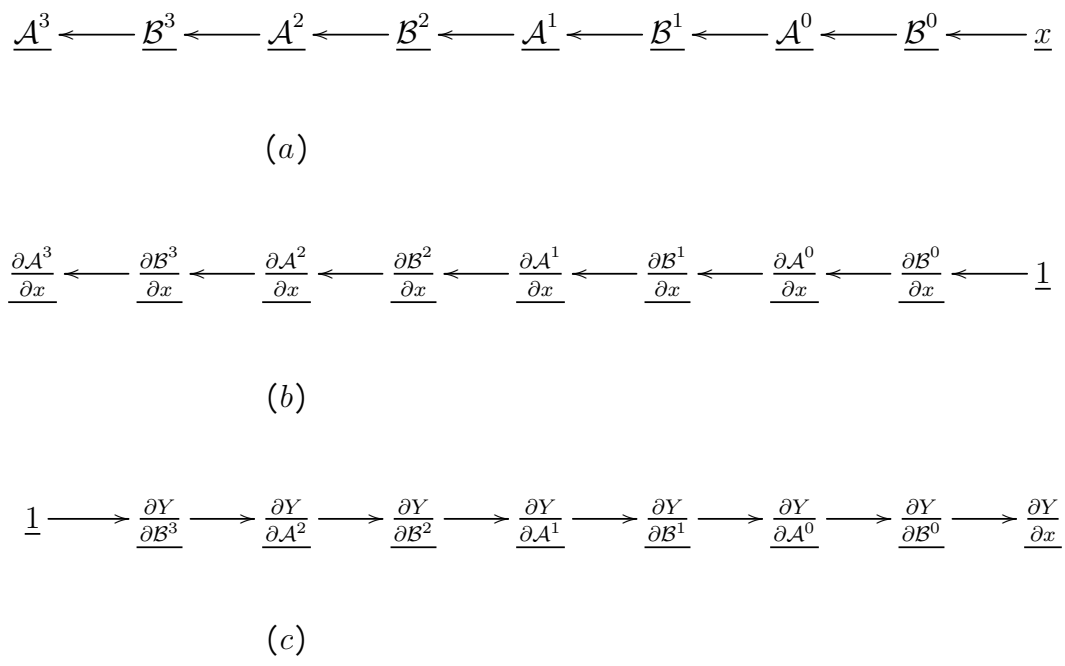


Figure 3.3: bnets for (a) function composition, (b) forward propagation and (c) back propagation for a neural net with 4 layers (3 hidden and output visible).

One has

$$\frac{\partial \mathcal{E}}{\partial w^{\lambda}_{i|j}} = \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} 2(Y_i - y_i[\sigma]) \frac{\partial Y}{\partial w^{\lambda}_{i|j}} \ . \tag{3.26}$$

Define $\mathcal{B}^{\lambda}_i$ thus

$$\mathcal{B}^{\lambda}_i(h^{\lambda-1}) = \sum_j w^{\lambda}_{i|j} h^{\lambda-1}_j \ . \tag{3.27}$$

Then

$$\frac{\partial Y}{\partial w^{\lambda}_{i|j}} = \frac{\partial Y}{\partial \mathcal{B}^{\lambda}_i} \frac{\partial \mathcal{B}^{\lambda}_i}{\partial w^{\lambda}_{i|j}} \tag{3.28}$$

$$= \frac{\partial Y}{\partial \mathcal{B}^{\lambda}_i} h^{\lambda-1}_j \tag{3.29}$$

$$\frac{\partial \mathcal{E}}{\partial w^{\lambda}_{i|j}} = \frac{\partial \mathcal{E}}{\partial \mathcal{B}^{\lambda}_{j}} \frac{\partial \mathcal{B}^{\lambda}_{j}}{\partial w^{\lambda}_{i|j}} \tag{3.30}$$

$$= \frac{\partial \mathcal{E}}{\partial \mathcal{B}^{\lambda}_{j}} h^{\lambda-1}_{j} \ . \tag{3.31}$$

This suggest that we can calculate the derivatives of the error $\mathcal{E}$ with respect to the weights $w^{\lambda}_{i|j}$ in two stages, using an intermediate quantity δ^{λ}_{j}:

$$\begin{cases} \delta^{\lambda}_{j} = \frac{\partial \mathcal{E}}{\partial \mathcal{B}^{\lambda}_{j}} \\ \frac{\partial \mathcal{E}}{\partial w^{\lambda}_{i|j}} = \delta^{\lambda}_{j} h^{\lambda-1}_{j} \end{cases} \tag{3.32}$$

To apply what we learned in the earlier General Theory section of this chapter, consider a NN with 4 layers (3 hidden, and the output visible one). Define the functions f_i as follows:

$$f^{0}_{i} = x_i \tag{3.33}$$

$$\text{Layer 0:} \quad f^{1}_{i} = \mathcal{B}^{0}_{i}(x_i), \quad f^{2}_{i} = \mathcal{A}^{0}_{i}(\mathcal{B}^{0}_{i}) \tag{3.34}$$

$$\text{Layer 1:} \quad f^{3}_{i} = \mathcal{B}^{1}_{i}(\mathcal{A}^{0}_{i}), \quad f^{4}_{i} = \mathcal{A}^{1}_{i}(\mathcal{B}^{1}_{i}) \tag{3.35}$$

$$\text{Layer 2:} \quad f^{5}_{i} = \mathcal{B}^{2}_{i}(\mathcal{A}^{1}_{i}), \quad f^{6}_{i} = \mathcal{A}^{2}_{i}(\mathcal{B}^{2}_{i}) \tag{3.36}$$

$$\text{Layer 3:} \quad f^{7}_{i} = \mathcal{B}^{3}_{i}(\mathcal{A}^{2}_{i}), \quad f^{8}_{i} = \mathcal{A}^{3}_{i}(\mathcal{B}^{3}_{i}) \tag{3.37}$$

See Fig.3.3. The TPMs, printed in blue, for the nodes of the bnet (c) for back propagation, are:

$$P\left(\frac{\partial Y}{\partial \mathcal{B}^{\lambda}} \mid \frac{\partial Y}{\partial \mathcal{B}^{\lambda+1}}\right) = \mathbb{1}\left(\frac{\partial Y}{\partial \mathcal{B}^{\lambda}} = \frac{\partial Y}{\partial \mathcal{B}^{\lambda+1}} \frac{\partial \mathcal{B}^{\lambda+1}}{\partial \mathcal{A}^{\lambda}} \frac{\partial \mathcal{A}^{\lambda}}{\partial \mathcal{B}^{\lambda}}\right) . \tag{3.38}$$

One has

$$\frac{\partial \mathcal{A}^{\lambda}_{i}}{\partial \mathcal{B}^{\lambda}_{j}} = D\mathcal{A}_i{}^{\lambda}(\mathcal{B}^{\lambda}_{i})\delta(i,j) \tag{3.39}$$

where $D\mathcal{A}^{\lambda}_{i}(z)$ is the derivative of $\mathcal{A}^{\lambda}_{i}(z)$.

From Eq.(3.27)

$$\mathcal{B}^{\lambda+1}_{i}(\mathcal{A}^{\lambda}) = \sum_{j} w^{\lambda+1}_{i|j} \mathcal{A}^{\lambda}_{j} \tag{3.40}$$

so

$$\frac{\partial \mathcal{B}^{\lambda+1}_{i}}{\partial \mathcal{A}^{\lambda}_{j}} = w^{\lambda+1}_{i|j} \ . \tag{3.41}$$

Therefore, Eq.(3.38) implies

$$P(\frac{\partial Y}{\partial \mathcal{B}_j^\lambda} \mid \frac{\partial Y}{\partial \mathcal{B}_j^{\lambda+1}}) = \mathbb{1}(\frac{\partial Y}{\partial \mathcal{B}_j^\lambda} = \sum_i \frac{\partial Y}{\partial \mathcal{B}_i^{\lambda+1}} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda) w_{i|j}^{\lambda+1}) \ , \tag{3.42}$$

$$P(\frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} \mid \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^{\lambda+1}}) = \mathbb{1}(\frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} = \sum_i \frac{\partial \mathcal{E}}{\partial \mathcal{B}_i^{\lambda+1}} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda) w_{i|j}^{\lambda+1}) \ , \tag{3.43}$$

$$\boxed{P(\delta_j^\lambda \mid \delta_j^{\lambda+1}) = \mathbb{1}(\delta_j^\lambda = \sum_i \delta_i^{\lambda+1} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda)) w_{i|j}^{\lambda+1})} \ . \tag{3.44}$$

First delta of iteration, belonging to output layer $\lambda = \Lambda - 1$:

$$\delta_j^{\Lambda-1} = \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^{\Lambda-1}} \tag{3.45}$$

$$= \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} 2(Y_i - y_i[\sigma]) D\mathcal{A}_i^{\Lambda-1}(\mathcal{B}_i^{\Lambda-1}) \delta(i,j) \tag{3.46}$$

$$= \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} 2(Y_j - y_j[\sigma]) D\mathcal{A}_j^{\Lambda-1}(\mathcal{B}_j^{\Lambda-1}) \tag{3.47}$$

Cute expression for derivative of sigmoid function:

$$D\mathrm{sig}(x) = \mathrm{sig}(x)(1 - \mathrm{sig}(x)) \tag{3.48}$$

Generalization to bnets (instead of Markov chains induced by layered structure of NNs):

$$P(\delta_{\underline{x}} \mid (\delta_{\underline{a}})_{\underline{a} \in ch(\underline{x})}) = \mathbb{1}(\delta_{\underline{x}} = \sum_{\underline{a} \in ch(\underline{x})} \delta_{\underline{a}} D\mathcal{A}_{\underline{x}}(\mathcal{B}_{\underline{x}})) w_{\underline{a}|\underline{x}}) \tag{3.49}$$

Reverse arrows of original bnet and define the TPM of nodes of "time reversed" bnet by

$$\boxed{P(\delta_{\underline{x}} \mid (\delta_{\underline{a}})_{\underline{a} \in pa(\underline{x})}) = \mathbb{1}(\delta_{\underline{x}} = \sum_{\underline{a} \in pa(\underline{x})} \delta_{\underline{a}} D\mathcal{A}_{\underline{x}}(\mathcal{B}_{\underline{x}})) w_{\underline{x}|\underline{a}}^T)} \tag{3.50}$$

Chapter 4

Basic Curve Fitting Using Gradient Descent

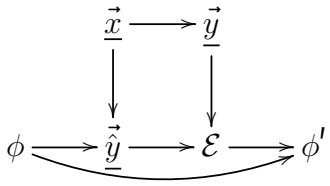


Figure 4.1: Basic curve fitting bnet.

Samples $(x[\sigma], y[\sigma]) \in S_{\underline{x}} \times S_{\underline{y}}$ are given. $nsam(\vec{x}) = nsam(\vec{y})$.
Estimator function $\hat{y}(x;\phi)$ for $x \in S_{\underline{x}}$ and $\phi \in \mathbb{R}$ is given.
Let

$$P_{\underline{x},\underline{y}}(x,y) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x = x[\sigma], y = y[\sigma]) \; . \tag{4.1}$$

Let

$$\mathcal{E}(\vec{x},\vec{y},\phi) = \frac{1}{nsam(\vec{y})} \sum_{\sigma} |y[\sigma] - \hat{y}(x[\sigma];\phi)|^2 \tag{4.2}$$

$\mathcal{E}$ is called the mean square error.

Best fit is parameters ϕ^* such that

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\, \mathcal{E}(\vec{x},\vec{y},\phi) \; . \tag{4.3}$$

The node TPMs for the basic curve fitting bnet Fig.4.1 are printed below in blue.

$$P(\phi) = \text{given} \; . \tag{4.4}$$

The first time it is used, ϕ is arbitrary. After the first time, it is determined by previous stage.

$$P(\vec{x}) = \prod_{\sigma} P_{\underline{x}}(x[\sigma]) \tag{4.5}$$

$$P(\vec{y}|\vec{x}) = \prod_{\sigma} P_{\underline{y}|\underline{x}}(y[\sigma] \mid x[\sigma]) \tag{4.6}$$

$$P(\hat{y}[\sigma]|\phi, \vec{x}) = \delta(\hat{y}[\sigma], \hat{y}(x[\sigma]; \phi)) \tag{4.7}$$

$$P(\mathcal{E}|\vec{\hat{y}}, \vec{y}) = \delta(\mathcal{E}, \frac{1}{nsam(\vec{x})} \sum_{\sigma} |y[\sigma] - \hat{y}[\sigma]|^2) \ . \tag{4.8}$$

$$P(\phi'|\phi, \mathcal{E}) = \delta(\phi', \phi - \eta \partial_{\phi} \mathcal{E}) \tag{4.9}$$

$\eta > 0$ is the descent rate. If $\Delta\phi = \phi' - \phi = -\eta \frac{\partial \mathcal{E}}{\partial \phi}$, then $\Delta\mathcal{E} = \frac{-1}{\eta}(\Delta\phi)^2 < 0$ so this will minimize the error $\mathcal{E}$. This is called "gradient descent".

Chapter 5

Bell and Clauser-Horne Inequalities in Quantum Mechanics

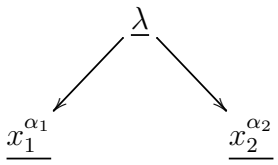
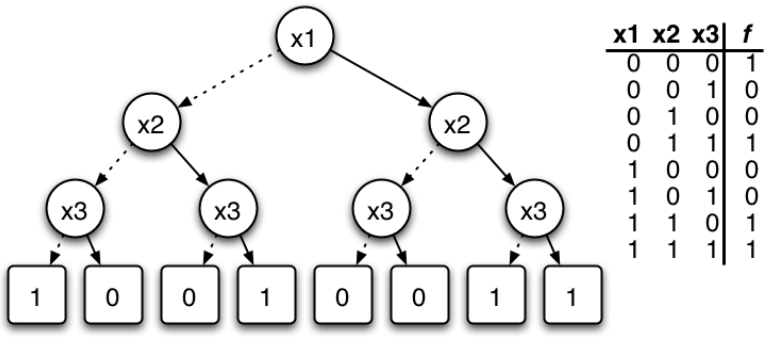


Figure 5.1: bnet used to discuss Bell and Clauser-Horne inequalities in Quantum Mechanics.

I wrote an article about this in 2008 for my blog "Quantum Bayesian Networks". See Ref.[29].

Chapter 6

Binary Decision Diagrams



x1	x2	x3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure 6.1: Binary decision tree and truth table for the function $f(x_1, x_2, x_3) = \bar{x}_1(x_2 + \bar{x}_3) + x_1x_2$

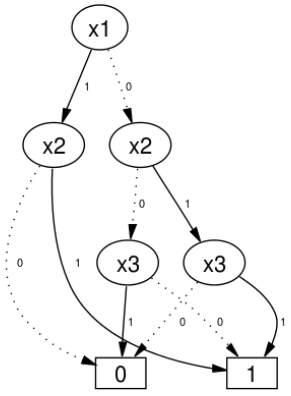


Figure 6.2: BDD for the function f of Fig.6.1.

This chapter is based on Wikipedia article Ref.[34].

Binary Decision Diagrams (BDDs) can be understood as a special case of Decision Trees (dtrees). We will assume that the reader has read Chapter 9 on dtrees before reading this chapter.

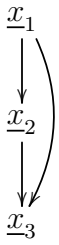
Both Figs.6.1 and 6.2 were taken from the aforementioned Wikipedia article. They give a simple example of a function $f : \{0,1\}^3 \to \{0,1\}$ represented in Fig.6.1 as a **binary decision tree** and in Fig.6.2 as a **binary decision diagram (BDD)**. The goal of this chapter is to find for each of those figures a bnet with the same graph structure.

We begin by noting that the function $f : \{0,1\}^3 \to \{0,1\}$ is a special case of a probability distribution $P : \{0,1\}^3 \to [0,1]$. In fact, if we restrict P to be deterministic, then $P_{det} : \{0,1\}^3 \to \{0,1\}$ has the same domain and range as f. Henceforth, we will refer to $f(x_1, x_2, x_3)$ as $P(x_1, x_2, x_3)$, keeping in mind that we are restricting our attention to deterministic probability distributions.

If we apply the chain rule for conditional probabilities to $P(x_1, x_2, x_3)$, we get

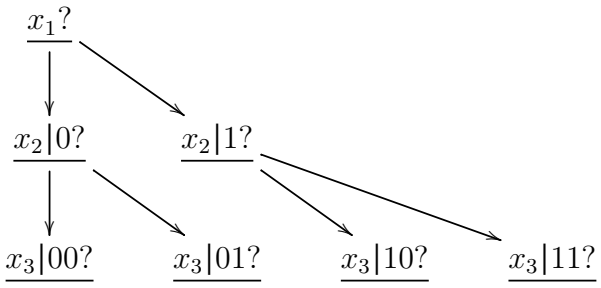
$$P(x_1, x_2, x_3) = P(x_3|x_1, x_2)P(x_2|x_1)P(x_1) \ , \tag{6.1}$$

which can be represented by the bnet:



(6.2)

But in Chapter 9, we learned how to represent the bnet of Eq.(6.2) as the bnet tree Eq.(6.3). In that tree, the nodes pose questions with 3 possible answers $0, 1, null$. In Eq.(6.3), $x_2|a?$ stands for "what is x_2 if $x_1 = a$?" and $x_3|a, b?$ stands for "what is x_3 if $x_1 = a, x_2 = b$?".



(6.3)

The node TPMs, printed in blue, for the bnet of Eq.(6.3) are as follows. If $x_1, x_2, x_3 \in \{0, 1, null\}$ and $a, b \in \{0, 1\}$, then

$$P(\underline{x_1?} = x_1) = \begin{cases} P_{\underline{x_1}}(x_1) & \text{if } x_1 \in \{0,1\} \\ 0 & \text{if } x_1 = null \end{cases} \tag{6.4}$$

$$P(\underline{x_2|a?} = x_2 \mid \underline{x_1?} = x_1) = \begin{cases} P_{\underline{x_2|x_1}}(x_2|a) & \text{if } x_1 = a \\ \mathbb{1}(x_2 = null) & \text{otherwise} \end{cases} \tag{6.5}$$

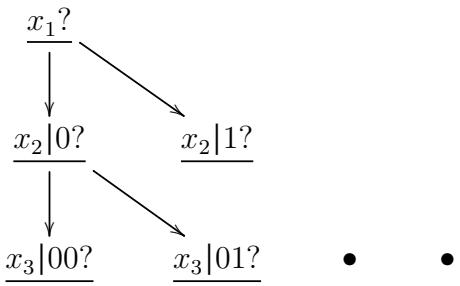
$$P(\underline{x_3|a,b?} = x_3 \mid \underline{x_2|b?} = x_2) = \begin{cases} P_{\underline{x_3}|\underline{x_1},\underline{x_2}}(x_3|a,b) & \text{if } (x_1,x_2) = (a,b) \\ \mathbb{1}(x_3 = null) & \text{otherwise} \end{cases} \tag{6.6}$$

The bnet shown in Eq.(6.3) contains the same info and has the same graph structure as the binary decision tree Fig.6.1. As when we were converting dtrees to their image bnets, the info in the endpoint nodes of Fig.6.1 is implicit in the node TPMs of the image bnet Eq.(6.3). If one wants to make the endpoint node info more explicit in the image bnet, one can add it to the descriptors of the state names of the leaf nodes of the image bnet. For example, one can add descriptors "gives $f = 0$" or "gives $f = 1$" to the "0" or "1" states of those leaf nodes.

The BDD shown in Fig.6.2 emphasizes the fact that

$$P(x_1, x_2, x_3|x_1 = 1) = P(x_2|x_1 = 1) = x_2 \ . \tag{6.7}$$

The BDD of Fig.6.2 corresponds to the bnet of Eq.(6.8).

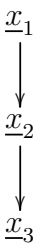


(6.8)

What happens if we consider an f for which $P(x_3|x_1,x_2) = P(x_3|x_2)$ so that one of the arcs of the fully connected bnet Eq.(6.2) is unnecessary? In that case,

$$P(x_1, x_2, x_3) = P(x_3|x_2)P(x_2|x_1)P(x_1) \ , \tag{6.9}$$

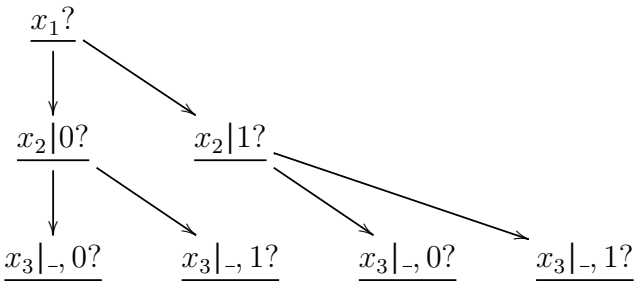
which can be represented by the Markov chain bnet:



(6.10)

Following the prescriptions of Chapter 9, we can represent the bnet of Eq.(6.10) as the bnet

tree Eq.(6.11). In that tree, the nodes pose questions with 3 possible answers $0, 1, null$.



(6.11)

Chapter 7

Chow-Liu Trees and Tree Augmented Naive Bayes (TAN)

This chapter is mostly based on chapter 8 of Pearl's 1988 book Ref.[18]. See also Ref.[37] and references therein.

This chapter uses various Shannon Information Theory entropies. Our notation for these entropies is described in Chapter 0.3 on Notational Conventions.

Chow-Liu trees refers to an algorithm for finding a bnet tree that fits an a priori given probability distribution as closely as possible.

Consider a bnet with n nodes $\underline{x}^n = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{n-1})$ such that $\underline{x}_i \in S_{\underline{x}_i}$ for all i. Let its total probability distribution be $P_{\underline{x}^n}$. For simplicity, we will abbreviate $P_{\underline{x}^n}$ by P. Hence

$$P(x^n) = P_{\underline{x}^n}(x^n) \ . \tag{7.1}$$

Suppose we want to fit $P_{\underline{x}^n}$ by a tree bnet with nodes $\underline{t}^n = (\underline{t}_0, \underline{t}_1, \ldots, \underline{t}_{n-1})$ such that $\underline{t}_i \in S_{\underline{t}_i} = S_{\underline{x}_i}$ for all i. For simplicity, we will abbreviate $P_{\underline{t}^n}$ by P_T. Hence

$$P_T(x^n) = P_{\underline{t}^n}(x^n) \ . \tag{7.2}$$

Throughout this chapter, let $V = \{0, 1, \ldots, n-1\}$, the set of vertices. Suppose μ is a function $\mu : V \to V$ such that $\mu(i) < i$. Let $T_\mu = \{\underline{t}_{\mu(i)} \to \underline{t}_i : i \in V - \{0\}\}$. Then T_μ is a tree that spans (i.e., it includes all nodes) $\underline{t}^n$. Its root node is $\underline{t}_0$, because $\underline{t}_0$ has no parents. All other nodes $\underline{t}_i$ have exactly one parent, namely $\underline{t}_{\mu(i)}$. Let P_T, the total probability distribution for the tree, be parameterized by the function μ as follows:

$$P_T(x^n) = \prod_{i=0}^{n-1} P_T(x_i | x_{\mu(i)}) \ , \tag{7.3}$$

where, for the root node 0, $P_T(x_0|x_{\mu(0)}) = P_T(x_0)$.

Claim 3 *$D_{KL}(P \parallel P_T)$ is minimized over all probability distributions P_T that are expressible as Eq.(7.3) iff*

$$P_T(x_i|x_{\mu(i)}) = P(x_i|x_{\mu(i)}) \tag{7.4}$$

for all i, and

$$\sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \tag{7.5}$$

is maximized over all μ.

proof:

$$\begin{aligned} D_{KL}(P \parallel P_T) &= \sum_{x^n} P(x^n) \ln \frac{P(x^n)}{P_T(x^n)} && (7.6)\\ &= -\sum_{x^n} \sum_i P(x^n) \ln P_T(x_i|x_{\mu(i)}) + \sum_i P(x^n) \ln P(x^n) && (7.7)\\ &= -\sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \ln P_T(x_i|x_{\mu(i)}) - H(\underline{x}^n) && (7.8)\\ &= -\sum_i \sum_{x_{\mu(i)}} P(x_{\mu(i)}) \left[\sum_{x_i} P(x_i|x_{\mu(i)}) \ln P_T(x_i|x_{\mu(i)}) \right] - H(\underline{x}^n) \ . && (7.9) \end{aligned}$$

Now note that

$$\sum_{x_i} P(x_i|x_{\mu(i)}) \ln \frac{P(x_i|x_{\mu(i)})}{P_T(x_i|x_{\mu(i)})} \geq 0 \tag{7.10}$$

and this inequality becomes an equality iff

$$P(x_i|x_{\mu(i)}) = P_T(x_i|x_{\mu(i)}) \ . \tag{7.11}$$

Therefore

$$D_{KL}(P \parallel P_T) \geq -\sum_i \underbrace{\sum_{x_{\mu(i)}} P(x_{\mu(i)}) \left[\sum_{x_i} P(x_i|x_{\mu(i)}) \ln P(x_i|x_{\mu(i)}) \right]}_{= H(\underline{x}_i|\underline{x}_{\mu(i)}) = H(\underline{x}_i : \underline{x}_{\mu(i)}) - H(\underline{x}_i)} - H(\underline{x}^n) \ , \tag{7.12}$$

and this inequality becomes an equality iff Eq.(7.11) is satisfied.

Note from the last equation that

$$\operatorname*{argmin}_\mu D_{KL}(P \parallel P_T) = \operatorname*{argmax}_\mu \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \ . \tag{7.13}$$

QED

Claim 4

$$\operatorname*{argmin}_\mu H(\underline{x}^n) = \operatorname*{argmax}_\mu \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \tag{7.14}$$

proof:

$$
\begin{aligned}
H(\underline{x}^n) &= -\sum_{x^n} P(x^n) \sum_i \ln P(x_i|x_{\mu(i)}) && (7.15)\\
&= -\sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \ln P(x_i|x_{\mu(i)}) && (7.16)\\
&= -\sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \left[\ln \frac{P(x_i|x_{\mu(i)})}{P(x_i)} + \ln P(x_i) \right] && (7.17)\\
&= -\sum_i \Big[H(\underline{x}_i : \underline{x}_{\mu(i)}) - H(\underline{x}_i) \Big] && (7.18)\\
&= \sum_i H(\underline{x}_i) - \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) && (7.19)
\end{aligned}
$$

QED

The meaning of Claims 3 and 4 is as follows. If $D_{KL}(P \parallel P_T)$ is minimized over all P_T, then

1. P_T inherits its TPM's from P, and
2. P_T gets its structure, which is being parameterized by the function μ, by maximizing the score given by

$$\text{score} = \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \ . \qquad (7.20)$$

(mutual information $H(\underline{a} : \underline{b})$ measures correlation between $\underline{a}$ and $\underline{b}$). Maximizing the score is the same as minimizing the entropy $H(\underline{x}^n)$ over all the structures μ. (i.e., finding least complex structure).

So far, we have studied the properties of those probability distributions P_T for a tree bnet that best approximates an a priori given probability distribution P, but we haven't yet described how to build a Chow-Liu tree based on empirical data. Next we give Chow-Liu's algorithm for doing so.

1. **Find MST using Kruskal's algorithm[1]. (see Fig.7.1)**
   Calculate weights $w_{i,j} = H(\underline{x}_i : \underline{x}_j)$ for all $i, j \in V$ and store them in a dictionary D that maps edges to weights.
   Order D by weight size.
   Let T be a list of the edges in the tree. Initialize T to empty.

[1] Kruskal's algorithm is one several famous algorithms (Prim's algo is another one) for finding an MST (maximum or minimum spanning tree). An MST algorithm takes an undirected graph with weights along its edges as input. It then finds a tree subgraph (i.e., subset of the edges of the graph with no loops) that (1) spans the graph (i.e., includes every vertex of the graph) and (2) maximizes (or minimizes) the sum of weights among all possible tree subgraphs. For more information, see Ref[53] and references therein, or any other of numerous explanations of MST in the Internet.

Repeat this until T has $n-1$ elements:

Remove largest weight w from D and corresponding edge e.

Add e to T if $\{e\} \cup T$ has no loops. Otherwise discard e and w.

2. **Give directions to edges in T. (see Fig.7.2)**

Let DT be a list of directed edges. Initialize DT to empty.

Choose any node as root node.

Point arrows along edges in T, away from root node.

Add new arrows to DT.

Repeat this until DT has $n-1$ elements:

Point arrows along edges in T, away from leaf nodes of current DT.

Add new arrows to DT.

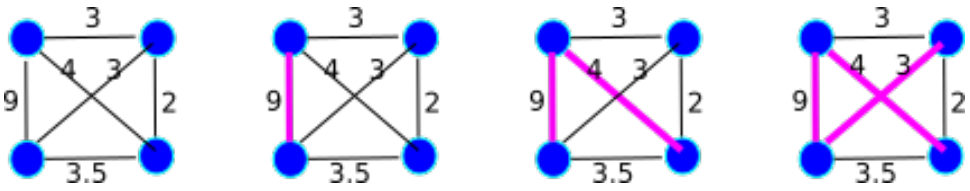


Figure 7.1: Example of finding MST (maximum spanning tree)

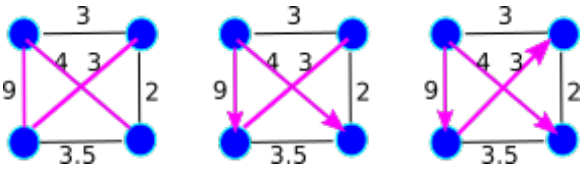


Figure 7.2: Example of giving directions to edges of spanning tree.

Nodes in a Chow-Liu tree can be rated in terms of their relative importance. Here are 2 possible metrics for measuring the importance of a node $\underline{a}$:

$$N_{nb}(\underline{a}) = \text{ number of neighbors of } \underline{a} \tag{7.21}$$

$$\text{traffic}(\underline{a}) = \sum_{\underline{n} \in nb(\underline{a})} H(\underline{a} : \underline{n}) \tag{7.22}$$

For example, to get a tree with low depth, one can choose as the root node the node which has largest N_{nb}, and if there are several with the same largest N_{nb}, choose out of those the one with the largest traffic.

Tree Augmented Naive Bayes (TAN)

Recall from Chapter 30 that a Naive Bayes bnet consists of a class node $\underline{c}$ with n children nodes $\underline{x}^n$, called the feature nodes. A Tree Augmented Naive Bayes (TAN) bnet is a Naive Bayes bnet with a tree grafted onto it like a chimera. More precisely, one starts with a Naive Bayes bnet and adds arrows between the feature nodes. The arrows are added in such a way that the TAN bnet sans node $\underline{c}$ constitutes a tree. It's not the most well motivated bnet in human history, but at least it adds a bit of correlation between the feature nodes of the Naive Bayes bnet. Those nodes are independent at fixed $\underline{c}$ in the Naive Bayes bnet, but are no longer so in the TAN bnet. See Figs.7.3 and 7.4 for an example of a TAN bnet.

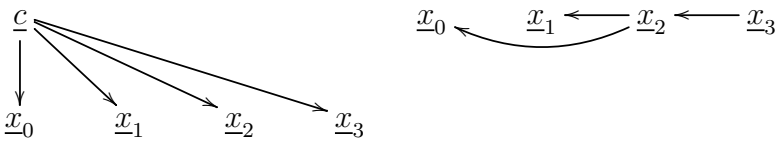


Figure 7.3: bnet for Naive Bayes with 4 feature nodes and another bnet for a tree made of the same feature nodes.

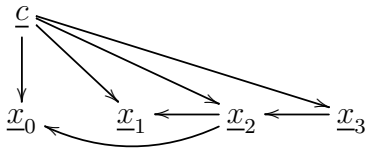


Figure 7.4: TAN bnet constructed by merging Naive Bayes bnet and tree bnet of Fig.7.3.

The total probability distribution P_{TAN} for a TAN bnet can be parameterized as follows.

$$P_{TAN}(x^n, c) = P_{TAN}(c) \prod_{i=0}^{n-1} P_{TAN}(x_i | x_{\mu(i)}, c) \ . \tag{7.23}$$

As with Chow Liu trees, we can attempt to find a TAN bnet whose total probability $P_{TAN} = P_{\underline{t}^n, \underline{c}}$ best approximates an a priori given probability distribution $P = P_{\underline{x}^n, \underline{c}}$.

Note that

Claim 5

$$\operatorname*{argmin}_{\mu} H(\underline{x}^n, \underline{c}) = \operatorname*{argmax}_{\mu} \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) \tag{7.24}$$

proof:

$$
\begin{aligned}
H(\underline{x}^n, \underline{c}) &= -\sum_{x^n, c} P(x^n, c) \left[\ln P(c) + \sum_i \ln P(x_i | x_{\mu(i)}, c) \right] && (7.25) \\
&= -\sum_{x^n, c} P(x^n, c) \left[\ln P(c) + \sum_i \ln \left(\frac{P(x_i, x_{\mu(i)} | c)}{P(x_i | c) P(x_{\mu(x_i)} | c)} P(x_i | c) \right) \right] && (7.26) \\
&= \sum_i H(\underline{x}_i, \underline{c}) - \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) && (7.27)
\end{aligned}
$$

QED

Following the same line of reasoning that we followed for Chow-Liu trees, we conclude that: If $D_{KL}(P \parallel P_{TAN})$ is minimized over all P_{TAN}, then

1. P_{TAN} inherits its TPM's from P, and

2. P_{TAN} gets its structure, which is being parameterized by the function μ, by maximizing the score defined by

$$
\text{score} = \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) \tag{7.28}
$$

One can build a TAN bnet from empirical data as follows:

Calculate a Chow-Liu Tree for each $c \in S_{\underline{c}}$. For each of those trees, create a TAN bnet, and calculate its score given by Eq.(7.28). Keep the TAN bnet with the largest score.

Chapter 8

Counterfactual Reasoning

This chapter is mostly based on Ref.[20], a 2019 review of causality by Pearl.

This chapter assumes that the reader has read Chapter 11 on do-calculus and Chapter 23 on LDEN (linear deterministic systems with external noise).

The 3 Rungs of Causal AI

According to Judea Pearl, there are 3 rungs in the ladder of causal AI. These are (as I see them):

1. **Observing Dumbly:** Collecting data and fitting curves to it, without any plan designed to investigate Nature's causal connections.

2. **Doing causal experiments:** Doing experiments consciously designed to elucidate Nature's causal connections. Even cats do this!, but current AI doesn't.

3. **Imagining counterfactual situations, Analogizing:** Imagining gedanken experiments to further understand Nature's causal connections, and to decide what future courses of action are more likely to succeed, even if there is zero direct data for those courses of action. Making predictions based on zero direct data is a very Bayesian concern, well out of the purview of frequentists. Nevertheless, humans do such "analogizing" all the time to great advantage. It becomes possible if there is some indirect but similar data that can be transported (transplanted, applied) to the situation of interest.

Chapter 27 on message passing is about rung 1. Chapter 11 on do-calculus is about rung 2. This chapter is dedicated to rung 3.

Two kinds of intervention operators

In Chapter 11, we introduced a **do operator** $\rho_{\underline{x}=x}$ (this is our notation for what Pearl symbolizes by $do(\underline{x}) = x$). The study of counterfactuals requires that we introduce a new kind of intervention operator that we will call an **imagine operator** and denote by $\kappa_{\underline{x}\to\underline{a}}(x)$.

The 2 types of intervention operators are defined graphically in Fig.8.1.

- The do operator $\rho_{\underline{x}=5}$ (called ρ because it turns $\underline{x}$ into a root node) amputates the incoming arrows of node $\underline{x}$ and sets the TPM of the new root node $\underline{x}$ to a delta function $\delta(x, 5)$ (or some state of $\underline{x}$ other than 5). Sometimes it is convenient, rather than calling the new node $\underline{x}$ like the old one, to call it by the new name $\rho\underline{x}$.

- The imagine operator $\kappa_{\underline{x}=5}$ (called κ because it creates konstant nodes) operates on arrows rather than nodes like the ρ operator does. $\kappa_{\underline{x}\to\underline{b}}(5)$ deletes arrow $\underline{x} \to \underline{b}$ and creates a new root node $\underline{x}'$ and a new arrow $\underline{x}' \to \underline{b}$. The TPM of the new node $\underline{x}'$ is a delta function $\delta(x', 5)$ (or some state of $\underline{x}$ other than 5). Sometimes it is convenient, rather than calling the new node $\underline{x}'$, to call it by the more explicit name $\kappa_{\underline{b}}\underline{x}$.

Now that we have both a do and an imagine operator, we realize, as Pearl did long ago, that we can create a **do-imagine-calculus** whose objective is to express probabilities such as $P(\underline{y}|\rho\underline{r} = r, \kappa_{\underline{b}}\underline{s} = s, t)$ in terms of observable probabilities that do not contain any do or imagine operators in them. As with do-calculus, this reduction is not always possible, and we say a probability is **identifiable** if it can be reduced in that manner. Such a do-imagine-calculus has already been developed by Pearl and collaborators, but we won't discuss it in this chapter (perhaps we will discuss it in a future one).

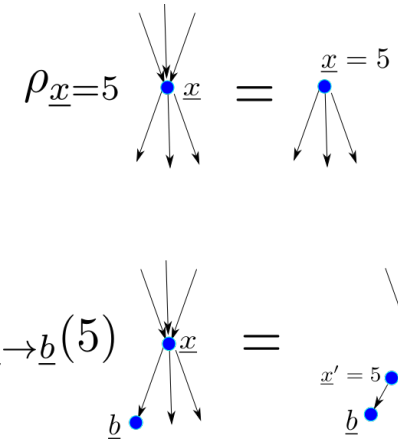


Figure 8.1: Action of "do" operator $\rho_{\underline{x}=5}$ on node $\underline{x}$ and of "imagine" operator $\kappa_{\underline{x}\to\underline{b}}(5)$ on arrow $\underline{x} \to \underline{b}$.

Do operator $\rho_{\underline{a}=a}$ for DEN diagrams

By the end of this chapter, the two kinds of intervention operators will be applied to DEN diagrams. Let us begin that journey by showing in this section how to apply the already familiar do operator to DEN diagrams.

Recall that the structural equations for a linear DEN, as given by Eq.(23.21) of Chapter 23, are:

$$\underline{x} = A\underline{x} + \underline{u} \ . \tag{8.1}$$

Therefore,

$$\underline{x} = (1 - A)^{-1}\underline{u} \tag{8.2}$$

which can be represented for both linear and non-linear DEN diagrams by:

$$\underline{x}_i = x_i(\underline{u}.) \tag{8.3}$$

If now we apply the operator $\rho_{\underline{a}=a}$ to the diagram described by the structural equations Eqs.8.1, we get the following new structural equations:

$$\underline{x}_i^* = \begin{cases} \sum_{j<i} A_{i|j}\underline{x}_j^* + u_i & \text{if } \underline{x}_i \neq \underline{a} \\ a & \text{if } \underline{x}_i = \underline{a} \end{cases} \ , \tag{8.4}$$

where we are calling $\underline{x}_i^*$ the nodes of the DEN diagram post intervention.

Eqs.(8.4) can be expressed in matrix notation as follows. Define $\pi_{\underline{a}}$ to be the $nx \times nx$ matrix with all entries equal to zero except for the (i_0, i_0) entry, which is 1. And define $e_{\underline{a}}$ to be the column vector with all entries zero except for the i_0'th one, which is 1. Here i_0 is defined so that $\underline{x}_{i_0} = \underline{a}$. In other words, $\pi_{\underline{a}}$ and $e_{\underline{a}}$ are defined by

$$(\pi_{\underline{a}})_{i,j} = \mathbb{1}(i = j, \underline{a} = \underline{x}_i) \tag{8.5}$$

and

$$(e_{\underline{a}})_i = \mathbb{1}(\underline{a} = \underline{x}_i) \ , \tag{8.6}$$

for $i, j \in \{0, 1, \ldots, nx - 1\}$. Next define

$$\pi_{!\underline{a}} = 1 - \pi_{\underline{a}} \ , \tag{8.7}$$

$$A^* = \pi_{!\underline{a}} A \ , \tag{8.8}$$

and

$$\underline{u}_{!\underline{a}} = \pi_{!\underline{a}}\underline{u} \ . \tag{8.9}$$

The effect of pre-multiplying the matrix A and the column vector $\underline{u}$ by $\pi_{!\underline{a}}$ is to leave all rows intact except for the i_0 row, which is set to zero. Here i_0 is defined by $\underline{a} = \underline{x}_{i_0}$. Finally, using all of the variables just defined, we can express the structural equations of the linear DEN diagram, post intervention, as

$$\underline{x}^* = A^*\underline{x}^* + \underline{u}_{!\underline{a}} + ae_{\underline{a}} \ . \tag{8.10}$$

Thus,

$$\underline{x}^* = (1 - A^*)^{-1}(\underline{u}_{!\underline{a}} + a e_{\underline{a}}) \; . \tag{8.11}$$

which can be represented for both linear and non-linear DEN diagrams by:

$$\underline{x}_i^* = x_i^*(\underline{u}_{!\underline{a}}, a) \; . \tag{8.12}$$

For any bnet,

$$P(\underline{y} = y | \underline{x} = x) = P_G(\underline{y} = y | \underline{x} = x) \tag{8.13}$$

$$P(\underline{y} = y | \rho\underline{x} = x) = P_{\rho_{\underline{x}=x} G}(\underline{y} = y) \tag{8.14}$$

Claim 6 *For a non-linear DEN diagram,*

$$P(y | \rho\underline{x} = x) = E\Big[\delta[y, y(\underline{u}_{!\underline{x}}, x)]\Big] \; . \tag{8.15}$$

proof:

$$\begin{aligned}
P(\underline{y} = y | \rho\underline{x} = x) &= P_{\rho_{\underline{x}=x} G}(\underline{y} = y) && (8.16)\\
&= \sum_{u_{!\underline{x}}} P(u_{!\underline{x}}) P_{\rho_{\underline{x}=x} G}(\underline{y} = y | u_{!\underline{x}}) && (8.17)\\
&= \sum_{u_{!\underline{x}}} P(u_{!\underline{x}}) \delta[y, y(u_{!\underline{x}}, x)] && (8.18)\\
&= E_{\underline{u}_{!\underline{x}}}[\delta[y, y(u_{!\underline{x}}, x)]] && (8.19)\\
&= E[\delta[y, y(\underline{u}_{!\underline{x}}, x)]] && (8.20)
\end{aligned}$$

QED

Claim 7 *For a nonlinear DEN diagram,*

$$E[\underline{y} | \rho\underline{x} = x] = E[y(\underline{u}_{!\underline{x}}, x)] \; . \tag{8.21}$$

proof:

$$\begin{aligned}
E[\underline{y} | \rho\underline{x} = x] &= \sum_y y P(\underline{y} = y | \rho\underline{x} = x) && (8.22)\\
&= \sum_y y E[\delta[y, y(u_{!\underline{x}}, x)]] && (8.23)\\
&= E[y(\underline{u}_{!\underline{x}}, x)] && (8.24)
\end{aligned}$$

QED

For any bnet

$$P(y|\rho\underline{x} = x, z) \;=\; \frac{P(y, z|\rho\underline{x} = x)}{P(z|\rho\underline{x} = x)} = P_{\rho_{\underline{x}=x}G}(y|x, z) \tag{8.25}$$

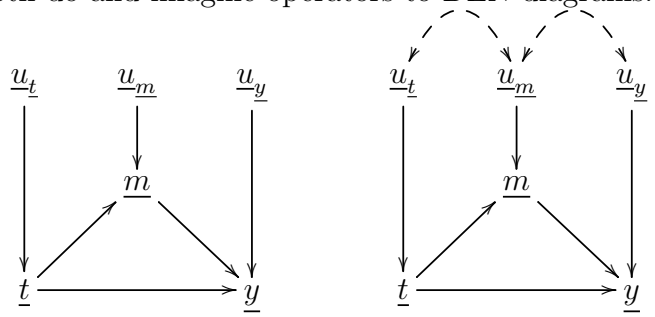
For a nonlinear DEN diagram,

$$P(y, z|\rho\underline{x} = x) = \sum_{u_{!\underline{x}}} P(u_{!\underline{x}})\delta[y, y(u_{!\underline{x}}, x)]\delta[z, z(u_{!\underline{x}}, x)] \tag{8.26}$$

$$P(z|\rho\underline{x} = x) = \sum_{u_{!\underline{x}}} P(u_{!\underline{x}})\delta[z, z(u_{!\underline{x}}, x)]\,. \tag{8.27}$$

Mediation Analysis

In the previous section, we applied the do operator to DEN diagrams. Mediation analysis is a nice example which applies both do and imagine operators to DEN diagrams.



G $\qquad\qquad$ G^*

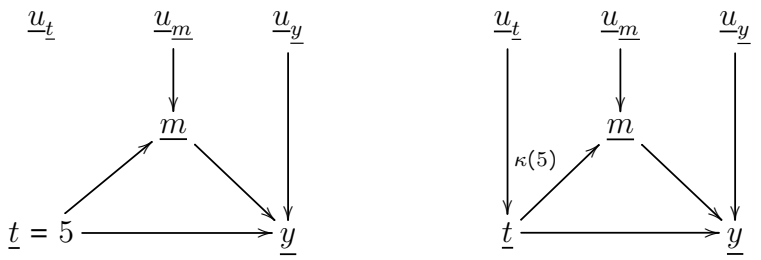
Figure 8.2: Graphs G and G^* are used to discuss mediation. In graph G, the exogenous variables are independent, whereas in graph G^* they are not.

The term "mediation analysis" refers to the analysis of the DEN diagram G in Fig.8.2. In that diagram, node $\underline{t}$ influences node $\underline{y}$ both directly and via the mediator node $\underline{m}$. The structural equations for that diagram are of the form:

$$\underline{t} \;=\; \underline{u}_{\underline{t}} \tag{8.28a}$$

$$\underline{m} \;=\; f_{\underline{m}}(\underline{t}, \underline{u}_{\underline{m}}) \tag{8.28b}$$

$$\underline{y} \;=\; f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_{\underline{y}})\,. \tag{8.28c}$$



$\rho_{\underline{t}=5}G$ $\kappa_{\underline{t}\to\underline{m}}(5)G$

Figure 8.3: Graph G of Fig.8.2 after applying do operator $\rho_{\underline{t}=5}$ and imagine operator $\kappa_{\underline{t}\to\underline{m}}(5)$.

Thus,

$$\underline{y} = f_{\underline{y}}(u_{\underline{t}}, f_{\underline{m}}(\underline{u}_{\underline{t}}, \underline{u}_{\underline{m}}), \underline{u}_{\underline{y}}) \ . \tag{8.29}$$

If we apply $\rho_{\underline{t}=5}G$ to Eqs.(8.28), we get

$$\begin{aligned} \underline{t} &= 5 && (8.30a)\\ \underline{m} &= f_{\underline{m}}(\underline{t}, \underline{u}_{\underline{m}}) && (8.30b)\\ \underline{y} &= f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_{\underline{y}}) \ . && (8.30c) \end{aligned}$$

Eqs.8.30 are represented graphically in Fig.8.3.

If we apply $\kappa_{\underline{t}\to\underline{m}}(5)G$ to Eqs.(8.28), we get

$$\begin{aligned} \underline{t} &= \underline{u}_{\underline{t}} && (8.31a)\\ \underline{m} &= f_{\underline{m}}(5, \underline{u}_{\underline{m}}) && (8.31b)\\ \underline{y} &= f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_{\underline{y}}) \ . && (8.31c) \end{aligned}$$

Eqs.8.31 are represented graphically in Fig.8.3.

Define the Total Effect (TE), and the Controlled Direct Effect (CDE) by

$$\begin{aligned} TE &= E[\underline{y}_{\rho_{\underline{t}=1}G} - \underline{y}_{\rho_{\underline{t}=0}G}] && (8.32)\\ CDE(m) &= E[\underline{y}_{\rho_{\underline{t}=1}\rho_{\underline{m}=m}G} - \underline{y}_{\rho_{\underline{t}=0}\rho_{\underline{m}=m}G}] && (8.33) \end{aligned}$$

The two DEN diagrams $\rho_{\underline{t}=t}G$ and $\rho_{\underline{t}=t}\rho_{\underline{m}=m}G$ used in the definitions of TE and CDE are given in Fig.8.4.

Let

$$E^b_a = E[\underline{y}_{\kappa_{\underline{t}\to\underline{y}}(a)\kappa_{\underline{t}\to\underline{m}}(b)G}] \tag{8.34}$$

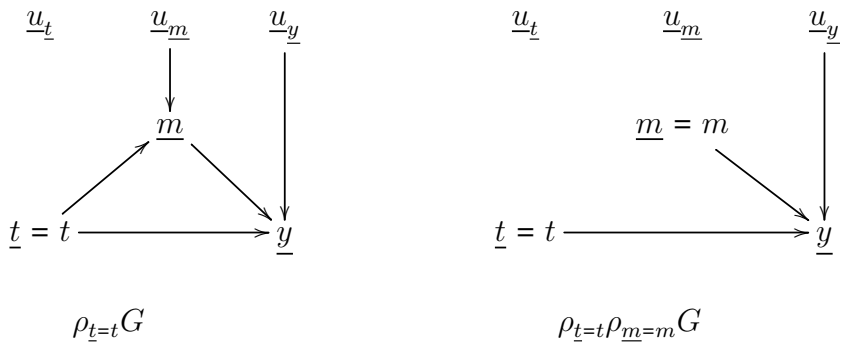


Figure 8.4: Graph G of Fig.8.2 after applying the do operators $\rho_{\underline{t}=t}$ and $\rho_{\underline{t}=t}\rho_{\underline{m}=m}$.

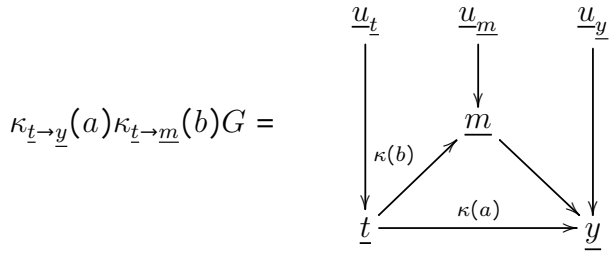


Figure 8.5: Graph G of Fig.8.2 after applying the imagine operator κ to arrows $\underline{t} \to \underline{m}$ and $\underline{t} \to \underline{y}$.

Fig.8.5 shows the diagram $\kappa_{\underline{t}\to\underline{y}}(a)\kappa_{\underline{t}\to\underline{m}}(b)G$ used in the definition of E^b_a.

Now define the Natural Direct Effect (NDE), and the Natural Indirect Effect (NIE) by

$$NDE = E^0_1 - E^0_0 \tag{8.35}$$
$$NIE(t) = E^1_t - E^0_t \ . \tag{8.36}$$

Note that

$$NDE + NIE(1) = (E^0_1 - E^0_0) + (E^1_1 - E^0_1) \tag{8.37}$$
$$= E^1_1 - E^0_0 \tag{8.38}$$
$$= TE \ . \tag{8.39}$$

Chapter 9

Decision Trees

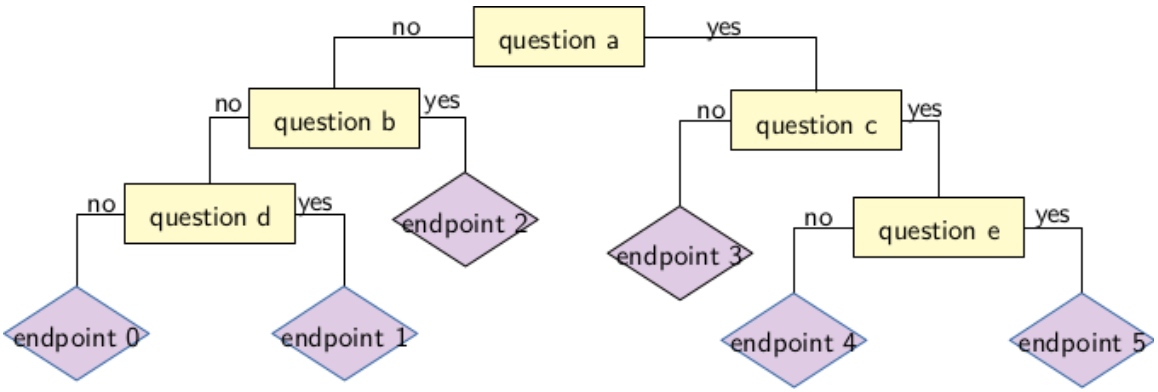


Figure 9.1: Typical decision tree.

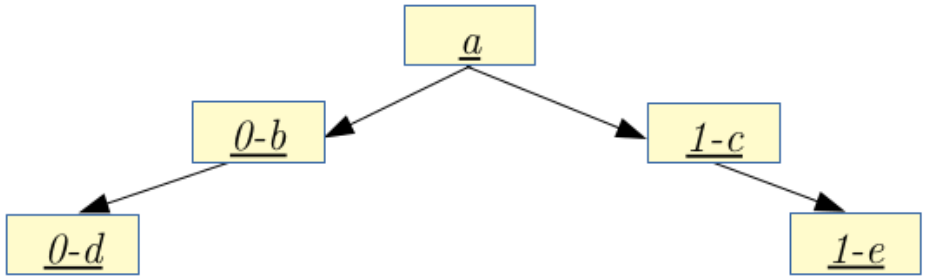


Figure 9.2: Bnet corresponding to decision tree Fig.9.1

Fig.9.1 shows a typical decision tree (dtree). The yellow rectangles pose questions. In general, the answers to those questions can be multiple choices with more than two choices, but in Fig.9.1 we have chosen the simplest case of only two choices, true or false. The purple diamonds represent endpoints, goals, final conclusions, single states of reality, etc.

A trivial observation that is often not made in dtree educational literature is that every dtree maps into a special bnet, let's call it its "image" bnet, in a very natural way. To get the image bnet, just follow the following simple steps:

1. **Keep the yellow question nodes but reinterpret them as bnet nodes. Reinterpret the connections among the dtree question nodes as arrows pointing down from the root node.**

   The image bnet nodes have 3 states, 0 = *no* and 1 = *yes* and *null*. Table 9.1 gives the node TPM $[P(x|a)]_{x\in\{0,1,null\},a\in\{0,1,null\}}$ where $p_1 \in [0,1]$ can be different for each node and is given in the info that specifies the dtree. In Table 9.1, $a_0 = 0$ if the dtree node being replaced has input "no" and $a_0 = 1$ if its input is "yes". $!a_0$ means not a_0 (i.e., $!a_0 = 1 - a_0$).

2. **This method of naming the image bnet nodes is not necessary but a good practice.** Give as name to each image bnet node an abridged version of the question that labels the dtree node it is replacing. Use as a suffix to the name of a bnet node either a 0 or a 1 depending whether the dtree node it is replacing has a 0 or a 1 as input. This suffix is not necessary because its info is already encoded into which column of the node TPM has zero probability for the *null* state, but it's a redundancy which makes the bnet easier to read and understand.

3. **Erase the purple endpoint nodes and connectors to them.** The info in each endpoint node can be preserved by storing it as a descriptor (e.g., tool tip) for the output states of the leaf node that is the parent to the endpoint in the image bnet. The endpoint info can be added to the descriptor of the *no* = 0 state if the endpoint has 0 as input or to the descriptor of the *yes* = 1 state if the endpoint has 1 as input.

$P(x\|a)$	$a = a_0$	$a = !a_0$	$a = null$
$x = 0$	$1 - p_1$	0	0
$x = 1$	p_1	0	0
$x = null$	0	1	1

Table 9.1: Transition probability matrix of a node of a dtree image bnet.

Table 9.2 describes the node types commonly used in dtrees.

When drawing dtrees, some people put info like explanations and probabilities on the connectors between the nodes of the dtree. That info can all be preserved in the TPM and the descriptors of the node names and node state names of the image bnet nodes. Often, the educational literature states that dtrees are more explicit and carry more info than their image bnets, but if one follows the above prescriptions, both can carry the same info.

A deterministic node commonly used in dtrees is one that asks the question $x < \alpha$?. for some real number $\alpha \in (L, U)$ and some variable x (for example, x = height of a person). For such an interval splitting node, the TPM would be as given in Table 9.3. If the interval $[L, U]$ is binned into a number *nbins* of bins, then this TPM will have dimensions (*nbins* + 1)×(the number of states of the parent node).

dtree node types (usual shape in parenthesis)	**their node TPM $P(x\|a)$ in image bnet**
chance node (oval)	$P(x\|a)$ arbitrary. random
decision node (square)	$P(x\|a) = \delta(x, f(a))$ where $f(\cdot)$ is a function of a. deterministic
endpoint node (diamond)	no $P(x\|a)$
fixed node	$P(x\|a) = \delta(x, x_0)$. x_0 does not depend on a whereas for decision node it does. deterministic.

Table 9.2: dtree node types.

$P(x\|a)$	states of parent node with $a = a_0$	states of parent node with $a =! a_0$	$a = null$
$[x \in bin]_{\forall\ bin \subset [L,\alpha)}$	1	0	0
$[x \in bin]_{\forall\ bin \subset [\alpha,U]}$	0	0	0
$x = null$	0	1	1

Table 9.3: Transition probability matrix for interval splitting node.

A naive Bayes bnet (see Chapter 30) consists of a single "class" node that fans out with arrows pointing to other "feature" nodes. If each leaf node of a naive Bayes bnet fans out into a set of new leaf nodes, and those new leaf nodes also fan out and so on recursively, we get a tree bnet. The bnet that arises from this recursive application of naive Bayes has the same graph structure as the image bnet of a dtree. However, it is more general because its node TPMs are more general; it has more weights (weight= parameters of node TPMs) than a dtree with the same graph.

Chapter 10

Digital Circuits

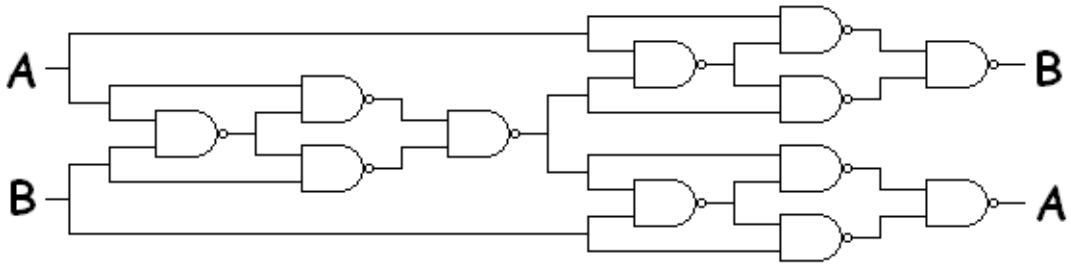


Figure 10.1: Typical digital circuit of NAND gates.

Digital (logic) gate: node with na input ports and nx output ports which represents a function

$$f : \{0,1\}^{na} \rightarrow \{0,1\}^{nx} \ . \tag{10.1}$$

Suppose
$a^{na} = (a_i)_{i=0,1,\ldots,na-1}$ where $a_i \in \{0,1\}$,
$x^{nx} = (x_i)_{i=0,1,\ldots,nx-1}$ where $x_i \in \{0,1\}$.
f maps a^{na} into x^{nx}.
Digital circuit (dcircuit) = circuit of digital gates.

Mapping any dcircuit to a bnet (Option A- See Fig.10.2)):

1. Replace every dcircuit gate described by Eq.(10.1) by nx bnet nodes $\underline{x}_i$ for $i = 0, 1, \ldots, nx-1$ such that

$$P(x_i|a^{na}) = \delta(x_i, f_i(a^{na})) \tag{10.2}$$

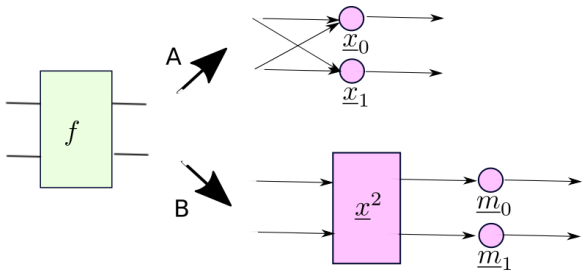


Figure 10.2: 2 options for mapping dcircuit node with multiple output ports into bnet.

2. Replace all connectors of the dcircuit by arrows pointing in the direction of the bit flow.

Mapping any dcircuit to a bnet (Option B- See Fig.10.2)):

1. Replace every dcircuit gate described by Eq.(10.1) with one bnet node called $\underline{x}^{nx}$ and, if $nx > 0$, nx "marginalizer nodes" $\underline{m}_i$ for $i = 0, 1, \ldots, nx-1$, such that

$$P(x^{nx}|a^{na}) = \delta(x^{nx}, f(a^{na})) \,, \tag{10.3}$$

and

$$P(m_i|x^{nx}) = \delta(m_i, x_i) \,. \tag{10.4}$$

2. Replace all connectors of the dcircuit by arrows pointing in the direction of the bit flow.

Options A and B don't work for digital circuits with feedback loops such as flip-flops. Those could probably be modeled with dynamical bnets.

Chapter 11

Do-Calculus

The do-calculus and associated ideas were invented by Judea Pearl and collaborators. This chapter is based on Judea Pearl's books. (See 0.4).

When doing do-calculus, it is convenient to separate the nodes of a bnet into 2 types: **visible (observed)**, and **non-visible (not observed, latent, hidden)**, depending on whether data describing the state of that node is available (visible) or not (non-visible). In this chapter, hidden nodes will be indicated in a bnet diagram by either: (1) enclosing their random variable in a box (as if it were inside a black box) or (2) making the arrows coming out of them dashed. Accordingly, the 3 diagrams in Fig.11.1 all mean the same thing.

A **confounder node for $\underline{x}$ and $\underline{y}$** (such as node $\underline{c}$ in Fig.11.1) is a hidden node with arrows pointing from it to both $\underline{x}$ and $\underline{y}$. In other words, it's an unobserved common cause of $\underline{x}$ and $\underline{y}$.

In this book, we will refer to a path all of whose nodes are observed as an **opath**.

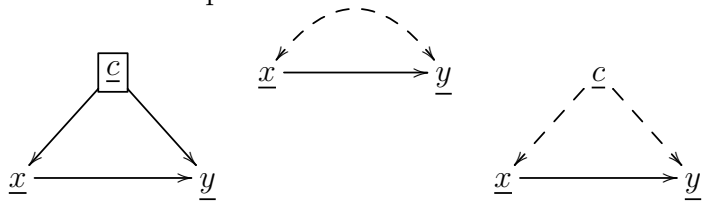


Figure 11.1: These 3 diagrams are equivalent. They mean that node $\underline{c}$ is hidden. Node $\underline{c}$ is implicit in the middle diagram.

Define an operator $\rho_{\underline{x}}$ that acts on a node $\underline{x}$ of a bnet to delete all the arrows entering $\underline{x}$, thus coverting $\underline{x}$ into a new node $\rho\underline{x}$ that is a root node. Define an analogous operator $\lambda\underline{x}$ that acts on a node $\underline{x}$ of a bnet to delete all the arrows leaving $\underline{x}$, thus converting $\underline{x}$ into a new node $\lambda\underline{x}$ that is a leaf node. $\rho_{\underline{x}}$ and $\lambda_{\underline{x}}$ are depicted in Fig.11.2.

If you don't know yet what we mean by a a multi-node $\underline{a}.$, see Chapter 0.2

Given a bnet G, we define as follows the operators $\rho_{\underline{a}.}$ and $\lambda_{\underline{a}.}$ for a multi-node $\underline{a}.$.

$$\rho_{\underline{a}.}G = \left[\prod_j \rho_{\underline{a}_j}\right] G \ , \quad \lambda_{\underline{a}.}G = \left[\prod_j \lambda_{\underline{a}_j}\right] G \ . \tag{11.1}$$

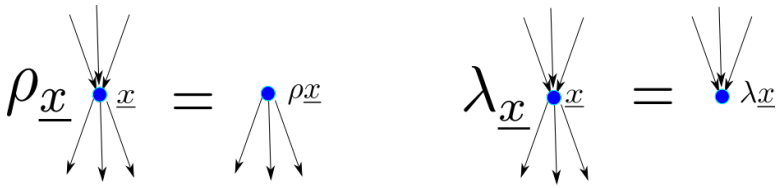


Figure 11.2: The operator $\rho_{\underline{x}}$ converts node $\underline{x}$ into a root node $\rho\underline{x}$. The operator $\lambda_{\underline{x}}$ converts node $\underline{x}$ into a leaf node $\lambda\underline{x}$.

Consider a bnet whose totality of nodes is labeled $\underline{X}.$. Recall that

$$P(X.) = \prod_j P(X_j|(X_k)_{k:\underline{X}_k \in pa(\underline{X}_j)}) \ . \tag{11.2}$$

Define an operator ρ that acts as follows[1]: Let $X. - a. = (X_k)_{k:\underline{X}_k \notin \underline{a}.}$.

$$\begin{aligned} P(X. - a.|\rho\underline{a}. = a.) &= \mathcal{N}(!(X. - a.)) \frac{P(X.)}{\prod_{j:\underline{X}_j \in \underline{a}.} P(X_j|(X_k)_{k:\underline{X}_k \in pa(\underline{X}_j)})} && (11.3) \\ &= \mathcal{N}(!(X. - a.)) \prod_{j:\underline{X}_j \notin \underline{a}.} P(X_j|(X_k)_{k:\underline{X}_k \in pa(\underline{X}_j)}) && (11.4) \\ &\neq P(X. - a.|\underline{a}. = a.) \ . && (11.5) \end{aligned}$$

Also,

$$P(X. - a., \rho\underline{a}. = a.') = P(X. - a.|\rho\underline{a}. = a.)\delta(a'., a.) \ . \tag{11.6}$$

In words, we replace the TPM for multinode $\underline{a}.$ by a deterministic prior distribution.

For instance, for the bnet

$$\underline{x} \longrightarrow \underline{y} \tag{11.7}$$

with

$$P(x, y) = P(y|x)P(x) \ , \tag{11.8}$$

one has

$$P(y|\rho\underline{x} = x) = P(y|x) \tag{11.9}$$

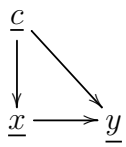
and

$$P(x|\rho\underline{y} = y) = P(x) \ . \tag{11.10}$$

[1] As usual, $\mathcal{N}(!x)$ denotes a constant that is independent of x.

This means that $\underline{x}$ causes $\underline{y}$ and $\underline{y}$ does not cause $\underline{x}$.

For the bnet



(11.11)

with

$$P(x, y, c) = P(y|x, c)P(x|c)P(c) \ , \tag{11.12}$$

one has

$$P(y, c|\rho\underline{x} = x) = P(y|x, c)P(c) \ . \tag{11.13}$$

Hence,

$$P(y|\rho\underline{x} = x) = \sum_c P(y|x, c)P(c) \ . \tag{11.14}$$

This is called **adjusting the parents of** $\underline{x}$.

For $\underline{b}. \subset \underline{X}. - \underline{a}.$, define

$$P(b.|\rho\underline{a}. = a.) = \sum_{X.-a.-b.} P(X. - a.|\rho\underline{a}. = a.) \ , \tag{11.15}$$

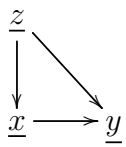
and for $\underline{s}. \subset \underline{X}. - \underline{a}. - \underline{b}.$, define

$$P(b.|\rho\underline{a}. = a., s.) = \frac{P(b., s.|\rho\underline{a}. = a.)}{P(s.|\rho\underline{a}. = a.)} \ . \tag{11.16}$$

$P(b.|\rho\underline{a}. = a., s.)$ is usually denoted instead by $P(b.|do(\underline{a}. = a.), s.)$. I prefer to use ρ instead of $do()$ to remind me that it generates root nodes. I'll still call ρ a **do operator**.

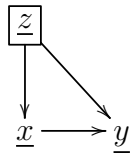
In $P(y|\rho\underline{x} = x)$, node $\underline{x}$ is turned into a root node. This guarantees that there is no confounding node connecting $\underline{x}$ and $\underline{y}$. Such confounding nodes are unwelcomed when calculating causal effects between the 2 variables $\underline{x}$ and $\underline{y}$ because they introduce non-causal correlations between the two. This is also what happens in a **Randomized Clinical Trial (RCT)**. In a RCT with treatment $\underline{x}$, the value of $\underline{x}$ for each patient is determined by a coin toss, effectively turning $\underline{x}$ into a root node. Hence, the do operator mimics a RCT.

$P(b.|\rho\underline{a}. = a., s.)$ is said to be **identifiable** if it can be expressed in terms of probability distributions that only depend on observed variables and that have no do operators in them. For example, $P(y|\rho\underline{x} = x)$ is identifiable for the bnet



(11.17)

but it is non-identifiable for the bnet



(11.18)

For $\underline{x}, \underline{y} \in \{0, 1\}$, the **causal effect difference** , or **average causal effect (ACE)** is defined as

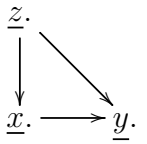
$$ACE = P(y = 1|\rho\underline{x} = 1) - P(y = 1|\rho\underline{x} = 0) \tag{11.19}$$

and the **Risk Difference (RD)** as

$$RD = P(y = 1|\underline{x} = 1) - P(y = 1|\underline{x} = 0) \ . \tag{11.20}$$

Parent Adjustment

Suppose that $\underline{x}., \underline{y}., \underline{z}.$ are disjoint multinodes and their union equals the totality of all nodes of a bnet. Suppose we have data available that allows us to estimate $P(x., y., z.)$. Hence, all nodes of the bnet are observable. Furthermore, suppose $\underline{z}. = pa(\underline{x}.)$. In other words, we are considering the bnet



. (11.21)

Then

$$P(y., z.|\rho\underline{x}. = x.) = P(y.|x., z.)P(z.) \tag{11.22}$$

so

$$P(y.|\rho\underline{x}. = x.) = \sum_{z.} P(y.|x., z.)P(z.) \ . \tag{11.23}$$

This is called **adjusting the parents** of $\underline{x}.$.

We say that we are **adjusting or controlling a variable** $\underline{a}$ if we condition a probability on $\underline{a}$ and then we average that probability over $\underline{a}$. More generally, we can adjust a whole multinode $\underline{a}.$ together.

Later on, we will introduce a generalization of this parent adjustment called the backdoor adjustment. In a backdoor adjustment, the adjusted multinode is not necessarily the parents of $\underline{x}.$, and $P(x., y., z.)$ need not represent the whole bnet.

3 Rules of do-calculus

Throughout this section, suppose $\underline{a}., \underline{b}., \underline{r}., \underline{s}.$ are disjoint multinodes in a bnet G.

Recall from Chapter 12 on d-separation, that $(\underline{b}. \perp_G \underline{a}.|\underline{r}., \underline{s}.)$ means that we have established from the d-separation rules that that all paths in G from $\underline{a}.$ to $\underline{b}.$ are blocked if we condition on $\underline{r}. \cup \underline{s}.$. Recall also that:

- **Rule 0:** Insertion or deletion of observations, without do operators. $(\underline{a}. = a. \leftrightarrow 1\ )$
  If $(\underline{b}. \perp_G \underline{a}.|\underline{r}., \underline{s}.)$, then $P(b.|a., r., s.) = P(b.|r., s.)$

The 3 rules of do-calculus can be presented in the same format.

- **Rule 1:** Insertion or deletion of observations $(\underline{a}. = a. \leftrightarrow 1\ )$
  If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\rho_{\underline{r}.}G$, then $P(b.|a., \rho\underline{r}. = r., s.) = P(b.|\rho\underline{r}. = r., s.)$.
- **Rule 2:** Action or observation exchange $(\rho\underline{a}. = a. \leftrightarrow \underline{a}. = a.)$
  If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\lambda_{\underline{a}.}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|a., \rho\underline{r}. = r., s.)$.
- **Rule 3:** Insertion and deletion of actions $(\rho\underline{a}. = a. \leftrightarrow 1)$
  If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\rho_{\underline{a}.-an(\underline{s}.)}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|\rho\underline{r}. = r., s.)$.

These rules have been proven to be sufficient for removing all do operators from an expression for which it is possible to do so.

Next we discuss two theorems that can be proven using do-calculus: the backdoor and the front-door adjustment theorems.

The backdoor theorem adjusts one multinode and the front-door theorem adjusts two.

Backdoor Adjustment

See Chapter 2 for examples of the use of the backdoor adjustment theorem. In this section, we shall mainly be concerned with proving this theorem using do-calculus.

Suppose we have access to data that allows us to estimate a probability distribution $P(x., y., z.)$. Hence, the variables $\underline{x}., \underline{y}., \underline{z}.$ are all observed (i.e, not hidden). Then we say that the backdoor $\underline{z}.$ satisfies the **backdoor adjustment criterion** relative to $(\underline{x}., \underline{y}.)$ if

1. All paths from $\underline{x}.$ to $\underline{y}.$ that start with an arrow pointing into $\underline{x}.$, are blocked by $\underline{z}.$.
2. $\underline{z}. \cap de(\underline{x}.) = \varnothing$.

Motivation for BD criterion: Part 1 rules out paths from $\underline{x}$ to $\underline{y}$ containing a fork node (confounder) which, if not blocked by $\underline{z}.$, would introduce a non-causal correlation (confounder bias). Part 2 rules out a directed path from $\underline{x}$ to $\underline{y}$ that has a mediator node blocked by $\underline{z}.$ or a collider node unblocked by $\underline{z}.$.

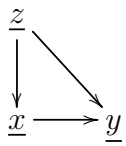
Claim 8 *Backdoor Adjustment Theorem*
If $\underline{z}.$ *satisfies the backdoor criterion relative to* $(\underline{x}., \underline{y}.)$*, then*

$$P(y.|\rho\underline{x}. = x.) = \sum_{z.} P(y.|x., z.)P(z.) \tag{11.24}$$

$$= \sum_{z.} \left\{ \begin{array}{l} \underline{z}. = z. \\ \\ \underline{x}. = x. \longrightarrow \underline{y}. \end{array} \right\} \tag{11.25}$$

proof:

For simplicity, let us omit the dots from the multinodes. If z satisfies the backdoor criterion relative to $(\underline{x}, \underline{y})$, then $\underline{x}, \underline{y}, \underline{z}$ must have the following structure.

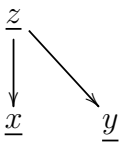


(11.26)

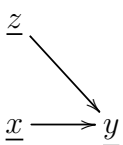
$P(y|\rho\underline{x} = x) =$

$= \sum_m P(y|\rho\underline{x} = x, z)P(z|\rho\underline{x} = x)$
by Probability Axioms

$= \sum_P (y|x, z)P(z|\rho\underline{x} = x)$
$P(y|\rho\underline{x} = x, z) \rightarrow P(y|x, z)$
by Rule 2: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\lambda_{\underline{a}.}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|a., \rho\underline{r}. = r., s.)$.
$\underline{y} \perp \underline{x}|\underline{z}$ in $\lambda_{\underline{x}}G$



$= \sum_z P(y|x, z)P(z)$
$P(z|\rho\underline{x} = x) \rightarrow P(z)$
by Rule 3: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\rho_{\underline{a}.-an(\underline{s}.)}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|\rho\underline{r}. = r., s.)$.
$\underline{z} \perp \underline{x}$ in $\rho_{\underline{x}}G$



(11.27)

QED

Note that the backdoor adjustment formula can be written as

$$P(y.|\rho\underline{x}. = x.) = \sum_{z.} P(y.|x., z.)P(z.) \tag{11.28}$$

$$= \sum_{z.} \frac{P(y., x., z.)}{P(x.|z.)} \tag{11.29}$$

This assumes $P(x.|z.) \neq 0$ for all $x., z.$. This assumption is referred to as **positivity**, and is violated if $P(x.|z.) = \delta(x., x.(z.))$. $P(x.|z.)$ is called the **propensity score** of $x.$ given $z.$. This equation does **inverse probability weighting**. One can approximate $P(x.|z.)$ in this equation to get an approximation to $P(y|\rho\underline{x} = x)$.

Front Door Adjustment

See Chapter 15 for examples of the use of the front-door adjustment theorem. In this section, we shall mainly be concerned with proving this theorem using do-calculus.

Suppose we have access to data that allows us to estimate a probability distribution $P(x., m., y.)$. Hence, the variables $\underline{x}., \underline{m}., \underline{y}.$ are all observed (i.e, not hidden). Then we say that the front-door $\underline{m}.$ satisfies the **front-door adjustment criterion** relative to $(\underline{x}., \underline{y}.)$ if

1. All directed paths from $\underline{x}.$ to $\underline{y}.$ are intercepted by (i.e., have a node in) $\underline{m}.$.
2. All backdoor paths from $\underline{x}.$ to $\underline{m}.$ are blocked.
3. All backdoor paths from on $\underline{m}.$ to $\underline{y}.$ are blocked by $\underline{x}.$.

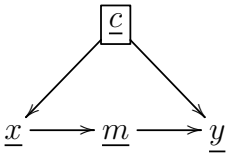
Claim 9 *Front-Door Adjustment Theorem*
If $\underline{m}.$ satisfies the front-door criterion relative to $(\underline{x}., \underline{y}.)$, and $P(x., m.) > 0$, then

$$P(y.|\rho\underline{x}. = x.) = \sum_{m.} \underbrace{\left[\sum_{x.'} P(y.|x'., m.)P(x'.)\right]}_{P(y.|\rho\underline{m}.=m.)} \underbrace{P(m.|x.)}_{P(m.|\rho\underline{x}.=x.)} \tag{11.30}$$

$$= \sum_{m.,x.'} \left\{ \underline{x}. = x.' \searrow \underline{x}. = x. \longrightarrow \underline{m}. = m. \longrightarrow \underline{y}. \right\} \tag{11.31}$$

proof: (See also Ref.[16] for a proof of the Front-Door Adjustment Theorem without using do-calculus.)

For simplicity, let us omit the dots from the multinodes. If $\underline{m}$ satisfies the front-door criterion relative to $(\underline{x}, \underline{y})$, then $\underline{x}, \underline{m}, \underline{y}$ must have the following structure, where node $\underline{c}$ is hidden.



(11.32)

Continues in next page.

$P(y|\rho\underline{x} = x) =$

$= \sum_m P(y|\rho\underline{x} = x, m)P(m|\rho\underline{x} = x)$

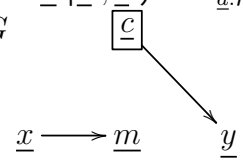
by Probability Axioms

$= \sum_m P(y|\rho\underline{x} = x, \rho\underline{m} = m)P(m|\rho\underline{x} = x)$

$P(y|\rho\underline{x} = x, m) \rightarrow P(y|\rho\underline{x} = x, \rho m = m)$

by Rule 2: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\lambda_{\underline{a}.}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|a., \rho\underline{r}. = r., s.)$.

$\underline{y} \perp \underline{m}|\underline{x}$ in $\lambda_{\underline{m}}\rho_{\underline{x}}G$

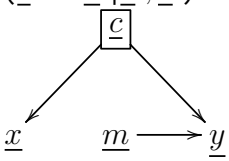


$= \sum_m P(y|\rho\underline{x} = x, \rho\underline{m} = m)P(m|x)$

$P(m|\rho\underline{x} = x) \rightarrow P(m|x)$

by Rule 2: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\lambda_{\underline{a}.}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|a., \rho\underline{r}. = r., s.)$.

$\underline{m} \perp \underline{x}$ in $\lambda_{\underline{x}}G$

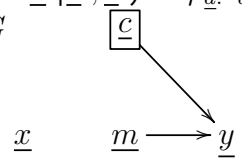


$= \sum_m P(y|\rho\underline{m} = m)P(m|x)$

$P(y|\rho\underline{x} = x, \rho\underline{m} = m) \rightarrow P(y|\rho\underline{m} = m)$

by Rule 3: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\rho_{\underline{a}.-an(\underline{s}.)}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|\rho\underline{r}. = r., s.)$.

$\underline{y} \perp \underline{x}|\underline{m}$ in $\rho_{\underline{x}}\rho_{\underline{m}}G$



$= \sum_{x'} \sum_m P(y|\rho\underline{m} = m, x')P(x'|\rho\underline{m} = m)P(m|x)$

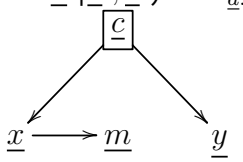
by Probability Axioms

$= \sum_{x'} \sum_m P(y|m, x')P(x'|\rho\underline{m} = m)P(m|x)$

$P(y|\rho\underline{m} = m, x') \rightarrow P(y|m, x')$

by Rule 2: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\lambda_{\underline{a}.}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|a., \rho\underline{r}. = r., s.)$.

$\underline{y} \perp \underline{m}|\underline{x}$ in $\lambda_{\underline{m}}G$

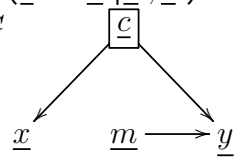


$= \sum_{x'} \sum_m P(y|m, x')P(x')P(m|x)$

$P(x'|\rho\underline{m} = m) \rightarrow P(x')$

by Rule 3: If $(\underline{b}. \perp \underline{a}.|\underline{r}., \underline{s}.)$ in $\rho_{\underline{a}.-an(\underline{s}.)}\rho_{\underline{r}.}G$, then $P(b.|\rho\underline{a}. = a., \rho\underline{r}. = r., s.) = P(b.|\rho\underline{r}. = r., s.)$.

$\underline{x} \perp \underline{m}$ in $\rho_{\underline{m}}G$



(11.33)

QED

Chapter 12

D-Separation

Before reading this chapter, I recommend that you read Chapter 0.2 on the definition of bnets.

A path γ that isn't a loop can have 3 types of intermediate nodes $\underline{x}$ (an intermediate node of γ is a node in γ that isn't one of the two end nodes). Suppose $\underline{a}$ and $\underline{b}$ are the two neighbors of $\underline{x}$. Then the 3 possible cases are:

1. **mediator node:** $(\underline{a} \leftarrow \underline{x} \leftarrow \underline{b})$ or $(\underline{a} \rightarrow \underline{x} \rightarrow \underline{b})$
2. **fork node:** $(\underline{a} \leftarrow \underline{x} \rightarrow \underline{b})$
3. **collider node:** $(\underline{a} \rightarrow \underline{x} \leftarrow \underline{b})$

We say that a non-loop path γ from $\underline{a}$ to $\underline{b}$ (i.e., with end nodes $\underline{a}, \underline{b}$) is **blocked** by a multinode $\underline{Z}.$ if one or more of the following statements is true:

1. There is a node $\underline{x} \in \underline{Z}.$ which is a mediator or a fork of γ.
2. γ contains a collider node $\underline{c}$ and $(\underline{c} \cup de(\underline{c})) \cap \underline{Z}. = \varnothing$ (i.e., neither $\underline{c}$ nor any of the descendants of $\underline{c}$ is contained in $\underline{Z}.$)

This definition of a blocked path is easy to remember if one thinks of the following analogy with pipes carrying a fluid. Think of path γ as if it were a pipe carrying a fluid. Think of the nodes of γ as junctions in the pipe. If $\underline{Z}.$ intersects γ at either a mediator or a fork junction, that blocks the pipe flow. A collider junction $\underline{c}$ is like a blackhole or a huge leak. Its presence blocks passage of the fluid as long as neither $\underline{c}$ nor any of the descendants of $\underline{c}$ are in $\underline{Z}.$. If, on the other hand, $\underline{c} \in \underline{Z}.$, or $\underline{c}' \in \underline{Z}.$ where $\underline{c}' \in de(\underline{c})$, then that acts as a complete (in the case of $\underline{c} \in \underline{Z}.$) or a partial (in the case of $\underline{c}' \in \underline{Z}.$) bridge across the blackhole.

See Fig.12.1 for some examples of paths that are blocked or not blocked by a multinode $\underline{Z}.$.

Given 3 disjoint multinodes $\underline{A}., \underline{B}., \underline{Z}.$ of a graph G, we write " $\underline{A}. \perp_G \underline{B}.|\underline{Z}.$" or say " $\underline{A}.$ **and** $\underline{B}.$ **are d-separated by** $\underline{Z}.$ **in** G" iff there exists no path γ from $\underline{a} \in \underline{A}.$, to $\underline{b} \in \underline{B}.$ which is not blocked by $\underline{Z}.$.

The minimal Markov blanket (see Chapter 24) of a node $\underline{a}$ is the smallest multinode $\underline{Z}.$ such that $\underline{a} \perp_G \underline{b}|\underline{Z}.$ for all $\underline{b} \notin \underline{a} \cup \underline{Z}.$.

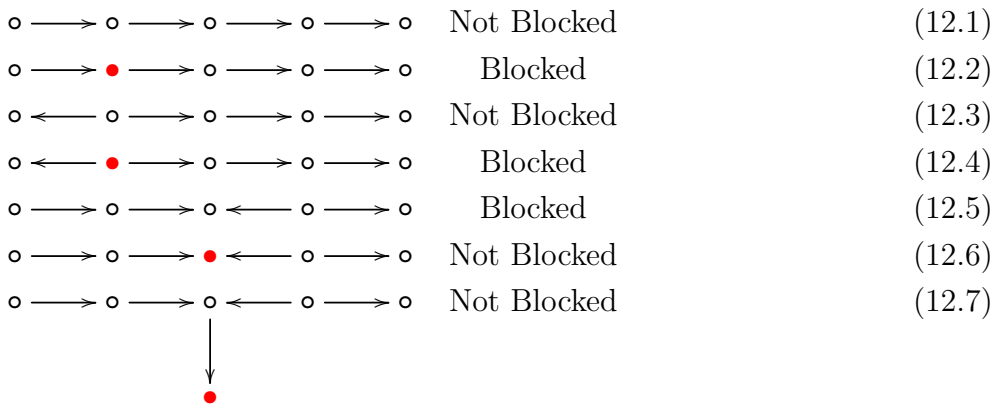


Figure 12.1: Examples of paths that are blocked or not blocked by a multinode $\underline{Z}.$. Nodes belonging to $\underline{Z}.$ are colored red.

We are finally ready to state the d-separation theorem, without proof.

A probability distribution P **is compatible with a DAG** G if P and G have the same random variables, and they can be combined to form a bnet without contradictions; i.e., one can calculate all the TPMs from P and multiply them together to obtain P again.

Claim 10 *(d-separation Theorem)*

Suppose $\underline{A}., \underline{B}., \underline{Z}.$ *are disjoint multinodes of a DAG* G.

If $\underline{A}. \perp_G \underline{B}.|\underline{Z}.$, *then* $P(B.|A., Z.) = P(B.|Z.)$ *for all* $B., A., Z.$, *for all* P *compatible with* G.

The full converse of the theorem can also be proven, but we won't be using it in this book.

Often, the right hand side of this theorem is stated as "$\underline{A}. \perp_P \underline{B}.|\underline{Z}.$ for all P". Then the theorem is stated: "If $\underline{A}. \perp_G \underline{B}.|\underline{Z}.$, then $\underline{A}. \perp_P \underline{B}.|\underline{Z}.$ for all P."

Note that the following are equivalent:

- $P(B.|A., Z.) = P(B.|Z.)$ for all $B., A., Z.$.
- $\underline{A}. \perp_P \underline{B}.|\underline{Z}.$
- $H(\underline{A}. : \underline{B}.|\underline{Z}.) = 0$ (see Chapter 0.3 for definition of conditional mutual information (CMI))

Extra stuff: mostly only for pure mathematicians

Below, we will use the notation $nde(\underline{a})$ to denote all nondescendants, including $\underline{a}$ itself, of a node $\underline{a}$ in a DAG G; i.e., all nodes of G that are not in $de(\underline{a}) \cup \underline{a}$, where $de(\underline{a})$ is defined in Chapter 0.2.

Given a DAG G, define the following sets of d-separations:[1]

$$DS(G) = \{(\underline{A}. \perp_G \underline{B}. \mid \underline{Z}.) : \ \underline{A}., \underline{B}., \underline{Z}. \text{ are multinodes of } G\} . \tag{12.8}$$

[1] Note that $(\underline{A}. \perp_G nde(\underline{A}.) \mid pa(\underline{A}.))$ and $(\underline{A}. \perp_G nde(\underline{A}.) - pa(\underline{A}.) \mid pa(\underline{A}.))$ are equivalent because $H(\underline{a} : \underline{b}, \underline{c}|\underline{c}) = H(\underline{a} : \underline{b}|\underline{c})$.

$$DS_{min}(G) = \{(\underline{A}.\ \perp_G nde(\underline{A}.) \mid pa(\underline{A}.)) : \ \underline{A}. \text{ is a multinode of } G\} \ . \tag{12.9}$$

See Chapter 34 for an example where set $DS_{min}(G)$ is calculated for a particular DAG G.

Claim 11 *For all DAGs G, $DS(G) = DS_{min}(G)$.*

Given a probability distribution P, define the following set of conditional independencies:

$$CI(P) = \{(\underline{A}.\ \perp_P \underline{B}. \mid \underline{Z}.) : \ \underline{A}., \underline{B}., \underline{Z}. \text{ are multinodes of } P\} \ , \tag{12.10}$$

For a DAG G and a probability distribution P compatible with G, define a map ϕ by

$$\phi : DS_{min}(G) \ \rightarrow \ CI(P) \tag{12.11}$$

$$\phi : \underline{A}.\ \perp_G nde(\underline{A}.) \mid pa(\underline{A}.) \ \mapsto \ \underline{A}.\ \perp_P nde(\underline{A}.) \mid pa(\underline{A}.) \tag{12.12}$$

In general, this map is 1-1 but not onto.

Claim 12 *For a bnet with a DAG G and a total probability distribution P, the map ϕ is a bijection.*

$DS(G)$ does not fully specify a DAG. DAGs with the same $DS(G)$ are said to be **d-separation equivalent**. See Chapter 34 for more info about d-separation equivalence.

Chapter 13

Dynamic Bayesian Networks

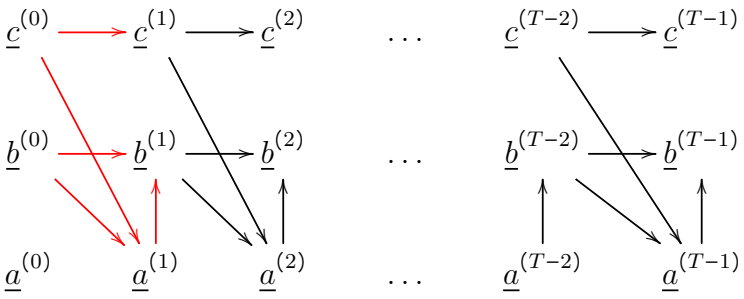


Figure 13.1: Example of a dynamic bnet. The pattern of red arrows is repeated $T-1$ times.

A dynamic bnet is simply a time homogeneous Markov chain (see Chapter 26) for which each node is called a **time slice**, and each time slice represents at finer resolution a sub-DAG which is the same between any 2 successive time slices. Fig.13.1 gives an example of a dynamic bnet. In Fig.13.1, we've drawn the 3 nodes of each time slice vertically, and labeled them with a superscript $.^{(t)}$, where $t \in \{0, 1 \ldots, T-1\}$ is the time of the slice. To fully specify the dynamic bnet of Fig.13.1, we would also have to specify the TPMs

$P(c^{(0)})$,
$P(b^{(0)})$
$P(c^{(1)}|c^{(0)})$,
$P(b^{(1)}|b^{(0)}, a^{(1)})$
$P(a^{(1)}|b^{(0)})$.

Dynamic bnets are very common in AI and Data Science. Kalman filters (Chapter 21), Hidden Markov Models (Chapter 18) and Recurrent Neural Networks (Chapter 36) are famous examples of dynamic bnets.

Bnets are acyclic; they can't have cycles (i.e, closed directed paths). Yet feedback loops are an important concept in Science. So what is the equivalent of feedback loops in the bnet world? Dynamic bnets are. Any feedback loop can be "unrolled" into a dynamic bnet.

Chapter 14

Expectation Maximization

This chapter is based on Refs.[40] and [63].

The Expectation Maximization (EM) algorithm is commonly used in Data Science to find the maximum over an **unknown parameter** θ of a likelihood function

$$P(\vec{x}|\theta) = \sum_{\vec{h}} P(\vec{x}, \vec{h}|\theta) \ , \tag{14.1}$$

where $\vec{x}$ denotes the **observed variables**, and $\vec{h}$ denotes the **latent variables**. Both θ and $\vec{h}$ are hidden (i.e., unobserved).[1]

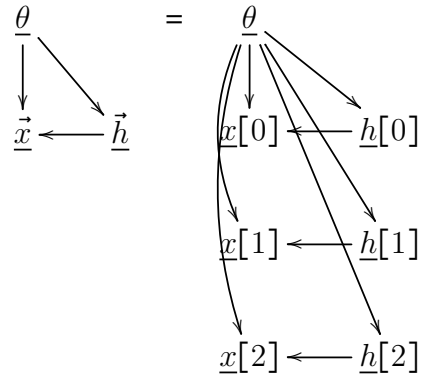


Figure 14.1: bnet for EM with $nsam = 3$.

The bnet for the EM algorithm is given by Fig.14.1 for $nsam = 3$. Later on in this chapter, we will give the node TPMs for this bnet for the special case in which $P(x[\sigma] \mid \theta)$ is a mixture (i.e., weighted sum) of Gaussians.

[1] The term "unknown parameter" is mainly of frequentist origin. For Bayesians, θ is a random variable with a delta function prior, whereas for frequentists, it is not a random variable at all, just an unknown parameter with no randomness.

Note that if we erase the $\underline{h}[\sigma]$ nodes from Fig.14.1, we get the bnet for naive Bayes, which is used for classification into the states of $\underline{\theta}$. However, there is one big difference. With naive Bayes, the leaf nodes have different TPMs. Here, we will assume they are i.i.d. Naive Bayes is used for classification: i.e., given the states of the leaf nodes, we infer the state of the root node. EM is used for clustering; i.e., given many i.i.d. samples, we fit their distribution by a weighted sum of prob distributions, usually Gaussians.

Let

$\mathcal{L}$ =likelihood function.

$nsam$ = number of samples.

$\vec{x} = (x[0], x[1], \ldots, x[nsam-1])$ $x[\sigma] \in S_{\underline{x}}$ for all σ.

$\vec{h} = (h[0], h[1], \ldots, h[nsam-1])$ $h[\sigma] \in S_{\underline{h}}$ for all σ.

We assume that the samples $(x[\sigma], h[\sigma])$ are i.i.d. for different σ at fixed θ. What this means is that there are probability distributions $P_{\underline{x}|\underline{h},\underline{\theta}}$ and $P_{\underline{h}|\underline{\theta}}$ such that

$$P(\vec{x}, \vec{h}|\theta) = \prod_{\sigma} \left[P_{\underline{x}|\underline{h},\underline{\theta}}(x[\sigma] \mid h[\sigma], \theta) P_{\underline{h}|\underline{\theta}}(h[\sigma] \mid \theta) \right] . \tag{14.2}$$

Definition of likelihood functions:

$$\underbrace{P(\vec{x}|\theta)}_{\mathcal{L}(\theta;\vec{x})} = \sum_{\vec{h}} \underbrace{P(\vec{x}, \vec{h}|\theta)}_{\mathcal{L}(\theta;\vec{x},\vec{h})} \tag{14.3}$$

θ^* = maximum likelihood estimate of θ (no prior $P(\theta)$ assumed):

$$\theta^* = \operatorname*{argmax}_{\theta} \mathcal{L}(\theta; \vec{x}) \tag{14.4}$$

The EM algorithm:

1. **Expectation step:**[2]

$$Q(\theta|\theta^{(t)}) = E_{\vec{h}|\vec{x},\theta^{(t)}} \ln P(\vec{x}, \vec{h}|\theta) \tag{14.5}$$

2. **Maximization step:**

$$\theta^{(t+1)} = \operatorname*{argmax}_{\theta} Q(\theta|\theta^{(t)}) . \tag{14.6}$$

Claim: $\lim_{t\to\infty} \theta^{(t)} = \theta^*$.

Fig.14.2 portrays the recursive nature of the EM algo as a dynamical, recurrent bnet. For Fig.14.2, the TPMs, printed in blue, for the $\underline{\theta}^{(t)}$ nodes for $t = 1, 2, \ldots$, are as follows:

$$P(\theta^{(t+1)}|\vec{x}, \theta^{(t)}) = \delta(\theta^{(t+1)}, \operatorname*{argmax}_{\theta} Q(\theta|\theta^{(t)})) . \tag{14.7}$$

---

[2] Note that that the right hand side of Eq.(14.5) is expressible in the form $\sum_{\sigma} \sum_{h[\sigma]} f(x[\sigma], h[\sigma])$.

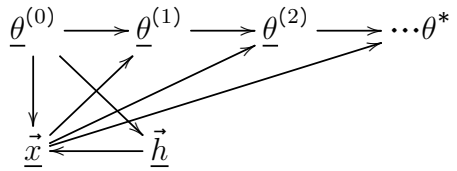


Figure 14.2: The EM algo generates a sequence of parameter estimates $(\theta^{(t)})_{t=0,1,2,...}$ that converges to the optimum (i.e., best-fit) parameter θ^*.

Motivation

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= E_{\vec{h}|\vec{x},\theta^{(t)}} \ln P(\vec{x},\vec{h}|\theta) && (14.8)\\
&= E_{\vec{h}|\vec{x},\theta^{(t)}}[\ln P(\vec{h}|\vec{x},\theta) + \ln P(\vec{x}|\theta)] && (14.9)\\
&= -D_{KL}\left(P(\vec{h}|\vec{x},\theta^{(t)}) \parallel P(\vec{h}|\vec{x},\theta)\right) - H[P(\underline{\vec{h}}|\vec{x},\theta^{(t)})] + \ln P(\vec{x}|\theta) && (14.10)
\end{aligned}
$$

When $\theta^{(t)} = \theta$, this becomes

$$Q(\theta|\theta) = -H[P(\underline{\vec{h}}|\vec{x},\theta)] + \ln P(\vec{x}|\theta)\ . \tag{14.11}$$

Hence,

$$
\begin{aligned}
\partial_\theta Q(\theta|\theta) &= -\sum_{\vec{h}} \partial_\theta P(\underline{\vec{h}}|\vec{x},\theta) + \partial_\theta \ln P(\vec{x}|\theta) && (14.12)\\
&= \partial_\theta \ln P(\vec{x}|\theta) && (14.13)
\end{aligned}
$$

So if $\theta^{(t)} \to \theta$ and $Q(\theta|\theta)$ is max at $\theta = \theta^*$, then $\ln P(\vec{x}|\theta)$ is max at $\theta = \theta^*$ too.

For a more rigorous proof that $\lim_{t\to\infty} \theta^{(t)} = \theta^*$, see Wikipedia article Ref.[40] and references therein.

Minorize-Maximize (MM) algorithms

A function $\mu(\theta|\theta^{(t)})$ **is said to minorize a target function** $\mathcal{L}(\theta)$ iff for all θ at fixed $\theta^{(t)}$, it satisfies the "$\mu \le \mathcal{L}$ property"

$$\mu(\theta|\theta^{(t)}) \le \mathcal{L}(\theta)\ , \tag{14.14}$$

and the "$\mu = \mathcal{L}$ property"

$$\mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)})\ . \tag{14.15}$$

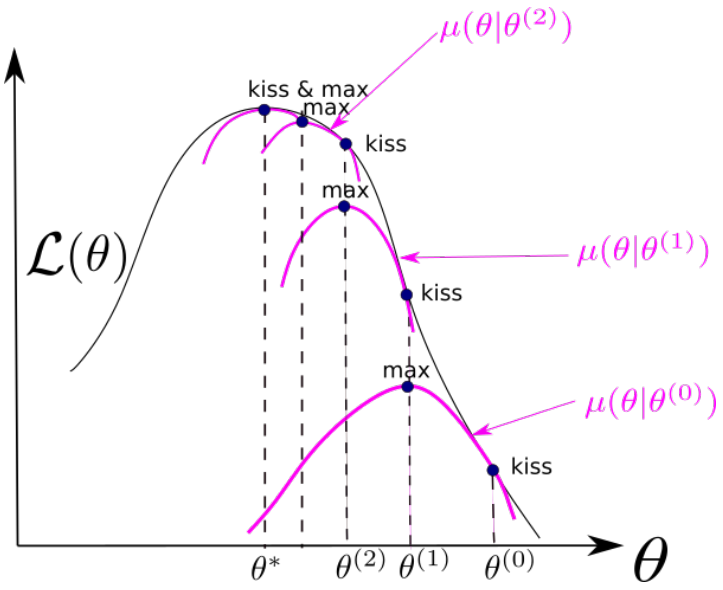


Figure 14.3: Function $\mu(\theta|\theta^{(t)})$ minorizes the function $\mathcal{L}(\theta)$. Note that $\mu(\theta|\theta^{(t)})$ is always below $\mathcal{L}(\theta)$. "max" indicates $\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, \mu(\theta|\theta^{(t)})$. "kiss" indicates $\mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)})$.

We **recursively maximize a minorizing function** $\mu(\theta|\theta^{(t)})$ if we define a sequence $(\theta^{(t)})_{t=0,1,...}$ as follows:

$$\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\, \mu(\theta|\theta^{(t)}) \,. \tag{14.16}$$

The sequence $(\mathcal{L}(\theta^{(t)}))_{t=0,1,2,...}$ generated by recursively maximizing a minorizing function must be nondecreasing:

$$\mathcal{L}(\theta^{(t+1)}) \geq \mu(\theta^{(t+1)}|\theta^{(t)}) \geq \mu(\theta^{(t)}|\theta^{(t)}) = \mathcal{L}(\theta^{(t)}) \,. \tag{14.17}$$

A **minorize-maximize (MM) algorithm** is any algo that specifies a minorizing function $\mu(\theta|\theta^{(t)})$ for a particular target function $\mathcal{L}(\theta)$. One can also define a **majorize-minimize algo (also called MM)** by inverting the inequalities throughout.

The EM algo is an MM algo. Indeed, if we define

$$\mathcal{L}(\theta) = \ln P(\vec{x}|\theta) \tag{14.18}$$

and

$$\mu(\theta|\theta^{(t)}) = Q(\theta|\theta^{(t)}) + H(P(\vec{\underline{h}}|\vec{x}, \theta^{(t)}) \,, \tag{14.19}$$

then Eq.(14.10) establishes the $\mu \leq \mathcal{L}$ and $\mu = \mathcal{L}$ properties required of a minorizing function.

How an MM algo works is portrayed in Fig.14.3.

Examples

Example (Gaussian mixture)

$x[\sigma] \in \mathbb{R}^d = S_{\underline{x}}$. $S_{\underline{h}}$ discrete and not too large. $n_{\underline{h}} = |S_{\underline{h}}|$ is number of Gaussians that we are going to fit the samples with.

Let

$$\theta = [w_h, \mu_h, \Sigma_h]_{h \in S_{\underline{h}}} , \tag{14.20}$$

where $[w_h]_{h \in S_{\underline{h}}}$ is a probability distribution of weights, and where $\mu_h \in \mathbb{R}^d$ and $\Sigma_h \in \mathbb{R}^{d \times d}$ are the mean value vector and covariance matrix of a d-dimensional Gaussian distribution.

The TPMs, printed in blue, for the nodes of Fig.14.1, for the special case of a mixture of Gaussians, are as follows:

$$P(x[\sigma] \mid h[\sigma] \mid \theta) = \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]}) \tag{14.21}$$

$$P(h[\sigma] \mid \theta) = w_{h[\sigma]} \tag{14.22}$$

Note that

$$\begin{aligned} P(x[\sigma] \mid \theta) &= \sum_h P(x[\sigma] \mid h[\sigma] = h, \theta) P(h[\sigma] = h \mid \theta) && (14.23) \\ &= \sum_h w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h) && (14.24) \end{aligned}$$

$$\begin{aligned} P(\vec{x}, \vec{h} | \theta) &= \prod_\sigma \left[w_{h[\sigma]} \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]}) \right] && (14.25) \\ &= \prod_\sigma \prod_h \left[w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h) \right]^{\mathbb{1}(h = h[\sigma])} && (14.26) \end{aligned}$$

Old Faithful: See Wikipedia Ref.[40] for an animated gif of a classic example of using EM to fit samples with a Gaussian mixture. Unfortunately, could not include it here because pdflatex does not support animated gifs. The gif shows samples in a 2 dimensional space (eruption time, delay time) from the Old Faithful geyser. In that example, $d = 2$ and $n_{\underline{h}} = 2$. Two clusters of points in a plane are fitted by a mixture of 2 Gaussians.

K-means clustering is often presented as the main competitor to EM for doing **clustering (non-supervised learning)**. In K-means clustering, the sample points are split into K mutually disjoint sets $S_0, S_1, \ldots, S_{K-1}$. The algorithm is easy to describe:

1. Initialize by choosing at random K data points $(\mu_k)_{k=0}^{K-1}$ called means or centroids and placing μ_k in S_k for all k.

2. **STEP 1:** For each data point, add it to the S_k whose centroid μ_k is closest to it.

3. **STEP 2:** Recalculate the centroids. Set μ_k equal to the mean value of set S_k.

4. Repeat steps 1 and 2 until the centroids stop changing by much.

Step 1 is analogous to the expectation step in EM, and Step 2 to the maximization step in EM (θ estimation versus μ_k estimation). We won't say anything further about K-means clustering because it isn't related to bnets in any way, and this is a book about bnets. For more info about K-means clustering, see Ref.[48].

Example (Blood Genotypes and Phenotypes):

Notation: $\vec{\underline{a}} = (\underline{a}[\sigma])_{\sigma=0,1,\ldots,nsam-1}$, where $nsam$ is the number of samples. Will sometimes denote $a[\sigma]$ by $a^{[\sigma]}$.

Suppose $\vec{\underline{x}} = (\vec{\underline{x}}_0)$ (i.e., just one component)

$\vec{\underline{h}} = (\vec{\underline{h}}_0)$ (i.e., just one component)

$\underline{h}[\sigma] \in S_{\underline{h}} = \{AA, AO, BB, BO, OO, AB\}$ (the 6 blood genotypes)

$\underline{x}[\sigma] \in S_{\underline{x}} = \{A, B, O, AB\}$ (the 4 blood phenotypes)

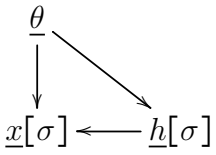


Figure 14.4: bnet for blood phenotypes $x[\sigma]$ and genotypes $h[\sigma]$.

For the bnet of Fig.14.4, the TPMs, printed in blue, are:

$$P(h^{[\sigma]}|\theta) = \begin{array}{c|c} AA & p_A^2 \\ AO & 2p_Ap_O \\ BB & p_B^2 \\ BO & 2p_Bp_O \\ OO & p_O^2 \\ AB & 2p_Ap_B \end{array} , \tag{14.27}$$

where $p_A + p_B + p_O = 1$.

$$P(x^{[\sigma]} \mid h^{[\sigma]}, \theta) = \begin{array}{c|cccccc} & AA & AO & BB & BO & OO & AB \\ A & 1 & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 1 & 1 & 0 & 0 \\ O & 0 & 0 & 0 & 0 & 1 & 0 \\ AB & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \tag{14.28}$$

$$\theta = (p_A, p_B) \tag{14.29}$$

Multiplying the TPMs in Eqs.(14.27 and (14.28), we get

$$P(x^{[\sigma]} \mid \theta) = \begin{array}{c|l} & \\ \hline A & p_A^2 + 2p_A p_O (= \pi_A) \\ B & p_B^2 + 2p_B p_O (= \pi_B) \\ O & p_O^2 (= \pi_O) \\ AB & 2p_A p_B (= \pi_{AB}) \end{array} \tag{14.30}$$

Note that

$$P(\vec{x}|\theta) = \prod_\sigma P(x^{[\sigma]}|\theta) \tag{14.31}$$

$$= (\pi_A)^{N_A} (\pi_B)^{N_B} (\pi_O)^{N_O} (\pi_{AB})^{N_{AB}} , \tag{14.32}$$

where N_x for $x \in S_{\underline{x}} = \{A, B, O, AB\}$ are the counts from the data. We can get estimates for the parameters p_A and p_B right here without doing EM. Just note that

$$\hat{\pi}_x = \frac{N_x}{N_+} \tag{14.33}$$

for $x \in S_{\underline{x}}$, where $N_+ = \sum_x N_x$. Eqs.(14.33) give 4 quadratic equations that can be solved for the parameters p_A, p_B in terms of the observed counts N_x for $x \in S_{\underline{x}}$.

If, instead, you want to find the optimum parameters p_A, p_B using EM, note that

$$Q(\theta|\theta^{(t)}) = \sum_{\vec{h}} P(\vec{h}|\theta^{(t)}) \ln P(\vec{x}, \vec{h}|\theta) \tag{14.34}$$

$$= \sum_{\vec{h}} \left[\prod_\sigma P(h^{[\sigma]}|\theta^{(t)}) \right] \ln \left[\prod_\sigma P(x^{[\sigma]}, h^{[\sigma]}|\theta) \right] \tag{14.35}$$

$$= \sum_\sigma \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(x^{[\sigma]}, h^{[\sigma]}|\theta) \tag{14.36}$$

$$= \sum_\sigma \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) [\ln P(x^{[\sigma]}|h^{[\sigma]}, \theta) + \ln P(h^{[\sigma]}|\theta)] \tag{14.37}$$

$$= nsam \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(h^{[\sigma]}|\theta) . \tag{14.38}$$

Example (Missing Data/Imputation):

The previous example on blood genotypes and phenotypes assumed no missing data in compiling the counts N_x. But what if there is missing data? Can one still apply the EM algo in that case? Yes! See Chapter 28.

Chapter 15

Front-door Adjustment

The front-door (FD) adjustment theorem is proven in Chapter 11 from the rules of do-calculus. The goal of this chapter is to give examples of the use of that theorem. We will restate the theorem in this chapter, sans proof. There is no need to understand the theorem's proof in order to use it. However, you will need to skim Chapter 11 in order to familiarize yourself with the notation used to state the theorem. This chapter also assumes that you are comfortable with the rules for checking for d-separation. Those rules are covered in Chapter 12.

Suppose we have access to data that allows us to estimate a probability distribution $P(x., m., y.)$. Hence, the variables $\underline{x}., \underline{m}., \underline{y}.$ are all observed (i.e, not hidden). Then we say that the front-door $\underline{m}.$ satisfies the **front-door adjustment criterion** relative to $(\underline{x}., \underline{y}.)$ if

1. All directed paths from $\underline{x}.$ to $\underline{y}.$ are intercepted by (i.e., have a node in) $\underline{m}.$.
2. All backdoor paths from $\underline{x}.$ to $\underline{m}.$ are blocked.
3. All backdoor paths from on $\underline{m}.$ to $\underline{y}.$ are blocked by $\underline{x}.$.

Claim 13 *Front-Door Adjustment Theorem*

If $\underline{m}.$ satisfies the front-door criterion relative to $(\underline{x}., \underline{y}.)$, and $P(x., m.) > 0$, then

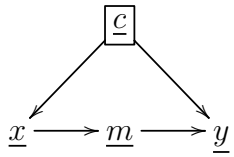
$$P(y.|\rho_{\underline{x}.} = x.) = \sum_{m.} \underbrace{\left[\sum_{x.'} P(y.|x'., m.) P(x'.) \right]}_{P(y.|\rho_{\underline{m}.}=m.)} \underbrace{P(m.|x.)}_{P(m.|\rho_{\underline{x}.}=x.)} \tag{15.1}$$

$$= \sum_{m.,x.'} \left\{ \begin{array}{ccccc} & & \underline{x}. = x.' & & \\ & & & \searrow & \\ \underline{x}. = x. & \longrightarrow & \underline{m}. = m. & \longrightarrow & \underline{y}. \end{array} \right\} \tag{15.2}$$

proof: See Chapter 11
QED

Examples

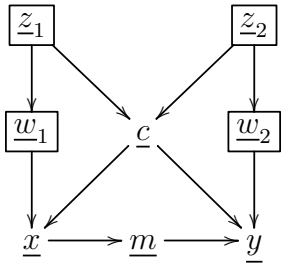
1.



(15.3)

If $\underline{x}. = \underline{x}, \underline{m}. = \underline{m}$ and $\underline{y}. = \underline{y}$, then the FD criterion is satisfied. Can't satisfy backdoor criterion because $\underline{z}.$ must be observed so can't block backdoor path $\underline{x} - \underline{c} - \underline{y}$.

2.



(15.4)

If $\underline{x}. = \underline{x}, \underline{m}. = \underline{m}$ and $\underline{y}. = \underline{y}$, then the FD criterion is satisfied. Can't satisfy backdoor criterion because to block backdoor path $\underline{x} - \underline{c} - \underline{y}$, need to condition on $\underline{c}$ (i.e., need $\underline{c} \in \underline{z}.$) but if this is true, then long path $\underline{x} - \underline{w}_1 - \underline{z}_1 - \underline{c} - \underline{z}_2 - \underline{w}_2 - \underline{y}$ becomes unblocked.

Chapter 16

Generative Adversarial Networks (GANs)

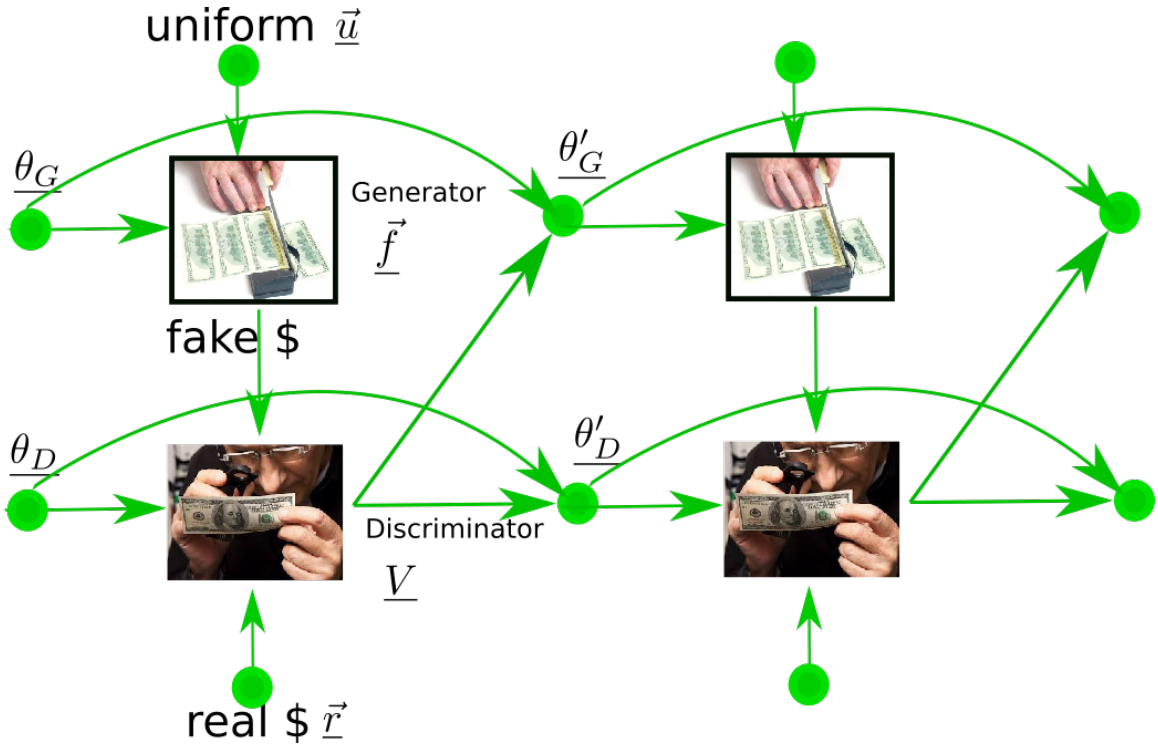


Figure 16.1: Generative Adversarial Network (GAN)

Original GAN, Ref.[5](2014).

Generator G (counterfeiter) generates samples $\vec{f}$ of fake money and submits them to Discriminator D (Treasury agent). D also gets samples $\vec{r}$ of real money. D submits veredict $V \in [0,1]$. G depends on parameter θ_G and D on parameter θ_D. Veredict V and initial θ_G, θ_D are used to get new parameters θ'_G, θ'_D.Process is repeated (Dynamical Bayesian Network) until saddle point in

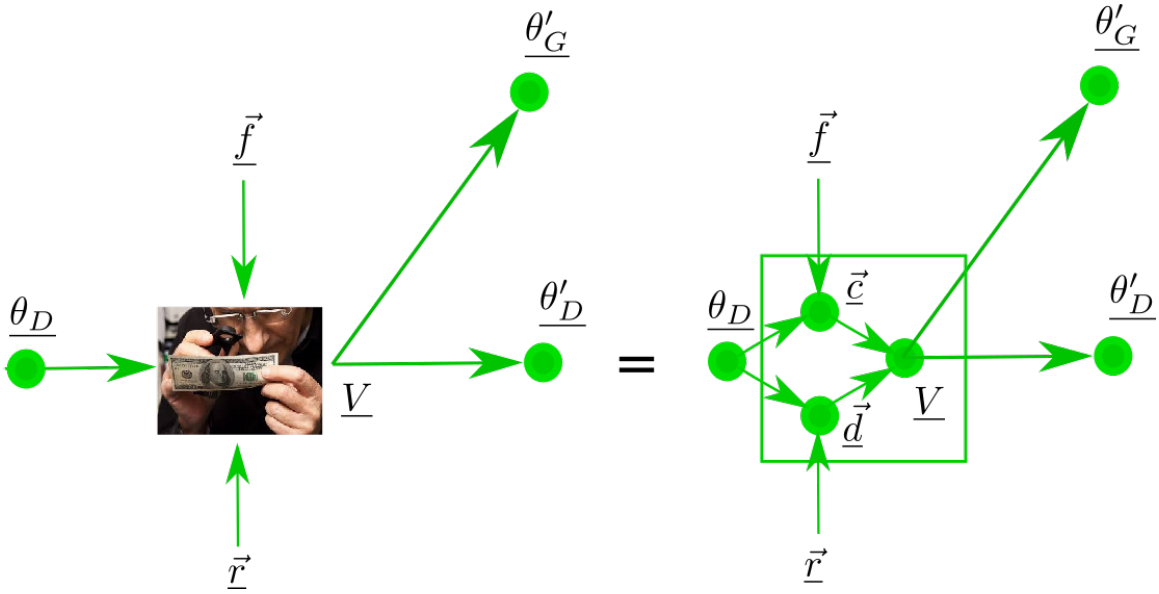


Figure 16.2: Discriminator node $\underline{V}$ in Fig.16.1 can be split into 3 nodes $\vec{\underline{c}}$, $\vec{\underline{d}}$ and $\underline{V}$.

$V(\theta_G, \theta_D)$ is reached. D makes G better and vice versa. Zero-sum game between D and G.
Let $\mathcal{D}$ be the domain of $D(\cdot, \theta_D)$. Assume that for any $x \in \mathcal{D}$,

$$0 \le D(x, \theta_D) \le 1 \ . \tag{16.1}$$

For any $S \subset \mathcal{D}$, define

$$\sum_{x \in S} D(x, \theta_D) = \lambda(S, \theta_D) \ . \tag{16.2}$$

In general, $G(\cdot, \theta_G)$ need not be real valued.
Assume that for every $u \in S_{\underline{u}}$, $G(u, \theta_G) = f \in S_{\underline{f}} \subset \mathcal{D}$. Define

$$\overline{D}(f, \theta_D) = 1 - D(f, \theta_D) \ . \tag{16.3}$$

Note that

$$0 \le \overline{D}(f, \theta_D) \le 1 \ . \tag{16.4}$$

Define:

$$V(\theta_G, \theta_D) = \sum_r P(r) \ln D(r, \theta_D) + \sum_u P(u) \ln \overline{D}(G(u, \theta_G), \theta_D) \ . \tag{16.5}$$

We want the first variation of $V(\theta_G, \theta_D)$ to vanish.

$$\delta V(\theta_G, \theta_D) = 0 \ . \tag{16.6}$$

This implies

$$\partial_{\theta_G} V(\theta_G, \theta_D) = \partial_{\theta_D} V(\theta_G, \theta_D) = 0 \tag{16.7}$$

and

$$V_{opt} = \min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) \ . \tag{16.8}$$

Node TPMs for Figs.16.1 and 16.2 are given next in blue:

$$P(\theta_G) = \text{ given} \tag{16.9}$$

$$P(\theta_D) = \text{ given} \tag{16.10}$$

$$P(\vec{u}) = \prod_i P(u[i]) \ \text{ (usually uniform distribution)} \tag{16.11}$$

$$P(\vec{r}) = \prod_i P(r[i]) \tag{16.12}$$

$$P(f[i] \mid \vec{u}, \theta_G) = \delta[f[i], G(u[i], \theta_G)] \tag{16.13}$$

$$P(c[i] \mid \vec{f}, \theta_D) = \delta(c[i], \overline{D}(f[i], \theta_D)) \tag{16.14}$$

$$P(d[j] \mid \vec{r}, \theta_D) = \delta(d[j], D(r[j], \theta_D)) \tag{16.15}$$

$$P(V|\vec{d}, \vec{c}) = \delta(V, \frac{1}{N} \ln \prod_{i,j} (c[i]d[j])) \tag{16.16}$$

where $N = nsam(\vec{r})nsam(\vec{u})$.

Let $\eta_G, \eta_D > 0$. Maximize V wrt θ_D, and minimize it wrt θ_G.

$$P(\theta'_G|V, \theta_G) = \delta(\theta'_G, \theta_G - \eta_G \partial_{\theta_G} V) \tag{16.17}$$

$$P(\theta'_D|V, \theta_D) = \delta(\theta'_D, \theta_D + \eta_D \partial_{\theta_D} V) \tag{16.18}$$

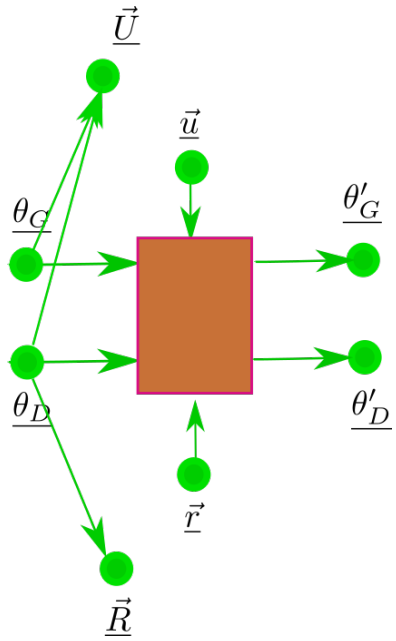


Figure 16.3: GAN, Constraining Bayesian Network

Constraining B net given in Fig.16.3. It adds 2 new nodes, namely $\underline{\vec{U}}$ and $\underline{\vec{R}}$, to the bnet of Fig.16.1. The purpose of these 2 barren (childrenless) nodes is to constrain certain functions to be probability distributions.

Node TPMs for the 2 new nodes given next in blue.

$$P(U[i] \mid \theta_G) = \frac{\overline{D}(G(U[i], \theta_G), \theta_D))}{\overline{\lambda}(\theta_G, \theta_D)} \tag{16.19}$$

where $S_{\underline{U[i]}} = S_{\underline{u}}$ and $\overline{\lambda}(\theta_G, \theta_D) = \sum_u \overline{D}(G(u, \theta_G), \theta_D))$.

$$P(R[i] \mid \theta_G, \theta_D) = \frac{D(R[i], \theta_D)}{\lambda(\theta_D)} \tag{16.20}$$

where $S_{\underline{R[i]}} = S_{\underline{r}}$ and $\lambda(\theta_D) = \sum_r D(r, \theta_D)$.

$$P(V|\vec{u}, \vec{r}) = \delta(V, \frac{1}{N} \ln \prod_{i,j} (P(\underline{R[i]} = r[i] \mid \theta_G, \theta_D) P(\underline{U[i]} = u[j] \mid \theta_G))) \tag{16.21}$$

where $N = nsam(\vec{r}) nsam(\vec{u})$.

$\mathcal{L}$ = likelihood

$$\mathcal{L} = P(\vec{r}, \vec{u}|\theta_G, \theta_D) \tag{16.22}$$

$$= \prod_{i,j} \left[\frac{D(r[i], \theta_D)}{\lambda(\theta_D)} \frac{\overline{D}(G(u[j], \theta_G), \theta_D))}{\overline{\lambda}(\theta_G, \theta_D)} \right] \tag{16.23}$$

$$\ln \mathcal{L} = N[V(\theta_G, \theta_D) - \ln \lambda(\theta_D) - \ln \overline{\lambda}(\theta_G, \theta_D)] \tag{16.24}$$

Chapter 17

Gaussian Nodes with Linear Dependence on Parents

Bnet nodes that have a Gaussian TPM with a linear dependence on their parent nodes (GLP) are a very popular way of modeling continuous nodes of bnets. A convenient aspect of them is that their parents can be discrete or continuous nodes, and their children can be discrete or continuous nodes too. Also, they can be learned easily from the data because their parameters can be expressed in terms of two node covariances. For these reasons, they are commonly used when doing structure learning of bnets with continuous nodes (see Chapter 42).

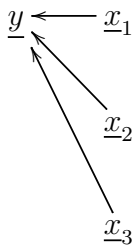


Figure 17.1: GLP node $\underline{y}$ with 3 parent nodes $\underline{x}^3 = (\underline{x}_1, \underline{x}_2, \underline{x}_3)$.

Recall our notation for a Gaussian distribution:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad , \tag{17.1}$$

where $x, \mu \in \mathbb{R}$ and $\sigma > 0$.

A GLP node $\underline{y}$ with n parents $\underline{x}^n = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n)$ has the following TPM:

$$P(y|x^n) = \mathcal{N}(y; \beta_0 + \beta^{nT} x^n, \sigma^2) \tag{17.2}$$

where $\underline{y}, \beta_0, \in \mathbb{R}$ and $\sigma^2 > 0$, and where $\underline{x}^n, \beta^n \in \mathbb{R}^n$ are **column vectors**. The T in β^{nT} stands for transpose. Any $\underline{x}_i$ can have a discrete set of states as long as they are real valued and ordinal (ordered by size). Fig.17.1 shows a diagrammatic representation of a GPL node with 3 parents.

Note that as $\sigma \to 0$, a GLP node becomes deterministic. In fact, it becomes a neural net node with a linear activation function.

An equivalent way of defining a GLP node $\underline{y}$ is in terms of a random variable equation expressing $\underline{y}$ as a hyperplane function of the parents $\underline{x}^n$ plus a Gaussian noise variable. Define an estimator $\underline{\hat{y}}$ of a "true value" $\underline{y}$ by

$$\underline{\hat{y}} = \beta_0 + \beta^{nT}\underline{x}^n \tag{17.3a}$$

and

$$\underline{y} = \underline{\hat{y}} + \underline{\epsilon} \tag{17.3b}$$

where the residual $\underline{\epsilon}$ satisfies

$$P(\epsilon) = \mathcal{N}(\epsilon; 0, \sigma^2) \tag{17.3c}$$

and

$$\langle \underline{x}^n, \underline{\epsilon} \rangle = 0 \; . \tag{17.3d}$$

The notation $\langle \underline{x}, \underline{y} \rangle$ for the covariance of random variables $\underline{x}$ and $\underline{y}$ is explained in Chapter 0.3.

Claim 14 *The parameters of a GLP node can be expressed in terms of 2-node covariances. Specifically,*

$$\beta^n = \langle \underline{x}^n, \underline{x}^{nT} \rangle^{-1} \langle \underline{y}, \underline{x}^n \rangle \tag{17.4}$$

$$\beta_0 = \langle \underline{y} \rangle - \beta^{nT} \langle \underline{x}^n \rangle \tag{17.5}$$

$$\sigma^2 = \langle \underline{y}, \underline{y} \rangle - \beta^{nT} \langle \underline{x}^n, \underline{y} \rangle \tag{17.6}$$

proof:

Note that $\langle \underline{x}^n, \underline{x}^{nT} \rangle^T = \langle \underline{x}^n, \underline{x}^{nT} \rangle$ and $\langle \underline{y}, \underline{x}^{nT} \rangle^T = \langle \underline{y}, \underline{x}^n \rangle$.

$$\langle \underline{y}, \underline{x}^{nT} \rangle = \beta^{nT} \langle \underline{x}^n, \underline{x}^{nT} \rangle \tag{17.7}$$

$$\langle \underline{y}, \underline{x}^n \rangle = \langle \underline{x}^n, \underline{x}^{nT} \rangle \beta^n \tag{17.8}$$

$$\beta^n = \langle \underline{x}^n, \underline{x}^{nT} \rangle^{-1} \langle \underline{y}, \underline{x}^n \rangle \tag{17.9}$$

$$\langle \underline{y} \rangle = \beta_0 + \beta^{nT} \langle \underline{x}^n \rangle \tag{17.10}$$

$$\langle \underline{y}, \underline{y} \rangle = \langle \beta_0 + \beta^{nT} \underline{x}^n + \underline{\epsilon}, \underline{y} \rangle \tag{17.11}$$

$$= \beta^{nT} \langle \underline{x}^n, \underline{y} \rangle + \sigma^2 \tag{17.12}$$

QED

Let D=Discrete, GLP=Gaussian with Linear dependence in Parents
The following arrows are possible in a bnet.

- $GLP \leftarrow GLP$
- $GLP \leftarrow D$

  Pass to GLP a separate set of regression coefficients β_0, β^n and variance σ^2 for each state of D. If D is called $\underline{d}$, let

$$P(y|(x^n)_d, d) = \mathcal{N}(y; (\beta_0)_d + (\beta^{nT})_d (x^n)_d, \sigma_d^2) \tag{17.13}$$

  for each $d \in S_{\underline{d}}$.

- $D \leftarrow GLP$

  If D expects a continuous parent, no need to preprocess GLP output. If D expects a discrete parent, break the interval $[a, b]$ that contains most of the range of the GPL node into subintervals and assign a discrete label to each subinterval.

- $D \leftarrow D$

Chapter 18

Hidden Markov Model

A Hidden Markov Model (HMM) is a generalization of a Kalman Filter (KF). KFs are discussed in Chapter 21. The bnets of HHMs and KFs bnets are the same. The only difference is that a KF assumes special node TPMs.

See Wikipedia article Ref.[44] to learn about the history and many uses of HMMs. This chapter is based on Ref.[15].

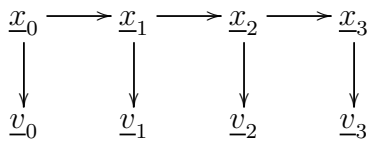


Figure 18.1: HMM bnet with $n = 4$.

Suppose
$\underline{v}^n = (\underline{v}_0, \underline{v}_1, \ldots, \underline{v}_{n-1})$ are n visible nodes that are measured, and
$\underline{x}^n = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{n-1})$ are the n hidden, unmeasurable state nodes of a system that is being monitored.

For the bnet of Fig.18.1, one has

$$P(x^n, v^n) = \prod_{i=0}^{n-1} P(x_i|x_{i-1})P(v_i|x_i) \ , \tag{18.1}$$

where $x_{-1} = 0$.

Let $x_{<i} = (x_0, x_1, \ldots, x_{i-1})$.
For $i = 0, 1, \ldots, n-1$, define
$\mathcal{F}_i$=future measurements probability

$$\mathcal{F}_i(x_i) = P(v_{>i}|x_i) \tag{18.2}$$

$\overline{\mathcal{F}}_i$= past and present measurements probability

$$\overline{\mathcal{F}}_i(x_i) = P(v_{<i}, v_i, x_i) \tag{18.3}$$

λ_i= present measurement probability

$$\lambda_i(x_i) = P(v_i|x_i) \tag{18.4}$$

$\mathcal{F}_i$, $\overline{\mathcal{F}}_i$ and λ_i can be represented graphically as follows:

$$\mathcal{F}_i(x_i) = \frac{1}{P(x_i)} \sum_{x_{>i}} \quad \begin{array}{ccc} x_i & \longrightarrow & x_{>i} \\ & & \downarrow \\ & & v_{>i} \end{array} \tag{18.5}$$

$$\overline{\mathcal{F}}_i(x_i) = \sum_{x_{<i}} \quad \begin{array}{ccc} x_{<i} & \longrightarrow & x_i \\ \downarrow & & \downarrow \\ v_{<i} & & v_i \end{array} \tag{18.6}$$

$$\lambda_i(x_i) = \frac{1}{P(x_i)} \quad \begin{array}{c} x_i \\ \downarrow \\ v_i \end{array} \tag{18.7}$$

Claim 15 *For $i \geq 0$,*

$$P(x_i, v^n) = \overline{\mathcal{F}}_i(x_i)\mathcal{F}_i(x_i) \ . \tag{18.8}$$

For $i > 0$,

$$P(x_{i-1}, x_i, v^n) = \overline{\mathcal{F}}_{i-1}(x_{i-1})\lambda_i(x_i)P(x_i|x_{i-1})\mathcal{F}_i(x_i) \ . \tag{18.9}$$

proof:

$$\begin{aligned} P(x_i, v^n) &= \sum_{x_{<i}} \sum_{x_{>i}} P(x^n, v^n) && (18.10)\\ &= \sum_{x_{<i}} \sum_{x_{>i}} P(x^n, v^n|x_i)P(x_i) && (18.11)\\ &= \sum_{x_{<i}} \sum_{x_{>i}} P(x_{<i}, v_{<i}, v_i|x_i)P(x_{>i}, v_{>i}|x_i)P(x_i) && (18.12)\\ &= P(v_{<i}, v_i|x_i)P(v_{>i}|x_i)P(x_i) && (18.13)\\ &= \overline{\mathcal{F}}_i(x_i)\mathcal{F}_i(x_i) && (18.14) \end{aligned}$$

$$
\begin{aligned}
P(x_{i-1}, x_i, v^n) &= \sum_{x_{<i-1}} \sum_{x_{>i}} P(x^n, v^n) && (18.15)\\
&= \sum_{x_{<i-1}} \sum_{x_{>i}} P(x^n, v^n | x_{i-1}, x_i) P(x_{i-1}, x_i) && (18.16)\\
&= \sum_{x_{<i-1}} \sum_{x_{>i}} P(x_{<i-1}, v_{<i-1}, v_{i-1} | x_{i-1}) P(v_i | x_i) P(x_{i-1}, x_i) P(x_{>i}, v_{>i} | x_i) && (18.17)\\
&= P(v_{<i-1}, v_{i-1} | x_{i-1}) P(v_i | x_i) P(x_{i-1}, x_i) P(v_{>i} | x_i) && (18.18)\\
&= \overline{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) && (18.19)
\end{aligned}
$$

QED

Claim 16 *For $i > 0$, $\mathcal{F}_i$ and $\overline{\mathcal{F}}_i$ can be calculated recursively as follows:*

$$\overline{\mathcal{F}}_i(x_i) = \sum_{x_{i-1}} \overline{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \tag{18.20}$$

$$\mathcal{F}_{i-1}(x_{i-1}) = \sum_{x_i} \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) \tag{18.21}$$

proof:

$$
\begin{aligned}
\overline{\mathcal{F}}_i(x_i) \mathcal{F}_i(x_i) &= P(x_i, v^n) && (18.22)\\
&= \sum_{x_{i-1}} P(x_{i-1}, x_i, v^n) && (18.23)\\
&= \sum_{x_{i-1}} \overline{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) && (18.24)
\end{aligned}
$$

$$
\begin{aligned}
\overline{\mathcal{F}}_{i-1}(x_{i-1}) \mathcal{F}_{i-1}(x_{i-1}) &= P(x_{i-1}, v^n) && (18.25)\\
&= \sum_{x_i} P(x_{i-1}, x_i, v^n) && (18.26)\\
&= \sum_{x_i} \overline{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) && (18.27)
\end{aligned}
$$

QED

Claim 17

$$P(x_i | x_{i-1}, v^n) = \frac{\lambda_i(x_i) \mathcal{F}_i(x_i)}{\mathcal{F}_{i-1}(x_{i-1})} P(x_i | x_{i-1}) \tag{18.28}$$

$$P(x_{i-1} | x_i, v^n) = \frac{\lambda_i(x_i) \overline{\mathcal{F}}_{i-1}(x_{i-1})}{\overline{\mathcal{F}}_i(x_i)} P(x_i | x_{i-1}) \tag{18.29}$$

proof:

$$P(x_i|x_{i-1}, v^n) = \frac{P(x_{i-1}, x_i, v^n)}{P(x_{i-1}, v^n)} \tag{18.30}$$

$$= \frac{\overline{\mathcal{F}}_{i-1}(x_{i-1})\lambda_i(x_i)P(x_i|x_{i-1})\mathcal{F}_i(x_i)}{\overline{\mathcal{F}}_{i-1}(x_{i-1})\mathcal{F}_{i-1}(x_{i-1})} \tag{18.31}$$

Analogous proof for Eq.(18.29).
QED

Chapter 19

Influence Diagrams & Utility Nodes

Influence diagrams are just arbitrary bnets enhanced with a new kind of node called an utility node. The rest of this brief chapter will be devoted to discussing utility nodes.

Suppose $U(x)$ is a deterministic function $U : S_{\underline{x}} \rightarrow \mathbb{R}$ called the **utility function**. Then the **expected utility** is defined as

$$E_{\underline{U}}[\underline{U}] = \sum_{U} P(U)U \tag{19.1}$$

$$= \sum_{x} \sum_{U} \underbrace{P(U|x)}_{\delta[U,U(x)]} P(x)U \tag{19.2}$$

$$= \sum_{x} P(x)U(x) \; . \tag{19.3}$$

An **utility node** can be understood as a node composed of 3 simpler bnet nodes. This is illustrated in Fig.19.1.

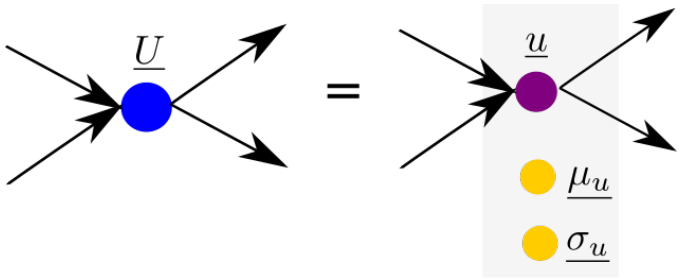


Figure 19.1: An utility node can be understood as a node composed of 3 simpler bnet nodes.

The TPMs, printed in blue, for the nodes of Fig.19.1, are as follows:

$$P(U|pa(U)) = \delta[U, U(pa(U))] \; , \tag{19.4}$$

where if $U : S_{\underline{x}} \rightarrow \mathbb{R}$, then $\underline{x} = pa(\underline{U})$.

$$P(u|pa(U)) = \delta[u, U(pa(U))] \tag{19.5}$$

Node $\underline{\mu}_u$ calculates the expected value (mean value) of $\underline{u}$:

$$P(\mu_u) = \delta(\mu_u, E_{\underline{u}}[\underline{u}]) \tag{19.6}$$

Node $\underline{\sigma}_u$ calculates the standard deviation of $\underline{u}$:

$$P(\sigma_u) = \delta(\sigma_U, \sqrt{E_{\underline{u}}[(\underline{u} - E_{\underline{u}}[\underline{u}])^2]}) \tag{19.7}$$

Note that in order to calculate expected values, it is necessary that $\underline{U}, \underline{u} \in \mathbb{R}$. Note that nodes $\underline{u}$, $\underline{\mu}_u$, $\underline{\sigma}_u$ must all 3 have access to the TPM $P(U|pa(U))$ of node $\underline{U}$. In fact, in order to calculate $E_{\underline{u}}[\cdot]$, it is necessary for nodes $\underline{\mu}_u$ and $\underline{\sigma}_u$ to have access not just to $P(U|pa(U))$ but also to $P(pa(U))$.

See Fig.19.2. An influence diagram may have multiple utility nodes ($\underline{U}_1$ and $\underline{U}_2$ in Fig.19.2). Then one can define a merging utility node $\underline{U}$ that sums the values of all the other utility nodes.

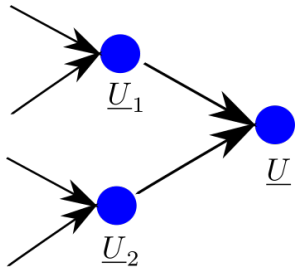


Figure 19.2: An influence diagram may have multiple utility nodes, say $\underline{U}_1$ and $\underline{U}_2$. Then one can define an utility node $\underline{U} = \underline{U}_1 + \underline{U}_2$.

For the node $\underline{U}$ of Fig.19.2,

$$P(U|U_1, U_2) = \delta(U, U_1 + U_2) \tag{19.8}$$

Chapter 20

Junction Tree Algorithm

The Junction Tree (JT) algorithm is an algo for evaluating exact marginals of a bnet, including cases in which some nodes are fixed to a single state. (fixed nodes are called the a priori evidence.)

The JT algo starts by clustering the loops of a bnet into bigger nodes so as to transform the bnet into a polytree bnet. Then it applies Pearl Belief Propagation (see Chapter 27) to the ensuing polytree. The first breakthrough paper to achieve this agenda in full was Ref.[8] by Lauritzen, and Spiegelhalter in 1988. See the Wikipedia article Ref.[47] for more info and references on the JT algorithm.

I won't describe the JT algo any further here, because it would take too long for this brief book to give a complete treatment of it, including the mathematical proofs. If all you want to do is to code the JT algo, without delving into the mathematical theorems and proofs behind it, I strongly recommend Ref.[7]. Ref.[7] is an excellent cookbook for programmers of the JT algo. My open source program QuantumFog (see Ref.[30]) implements the JT algo in Python, following the recipe of Ref.[7].

Chapter 21

Kalman Filter

A Kalman Filter is a special case of a Hidden Markov Model. HMMs are discussed in Chapter 18.

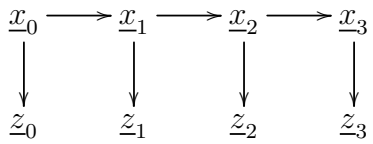


Figure 21.1: Kalman Filter bnet with $T = 4$.

Let $t = 0, 1, 2, \ldots, T-1$.
$\underline{x}_t \in S_{\underline{x}}$ are random variables that represent the hidden (unobserved) true state of the system.
$\underline{z}_t \in S_{\underline{z}}$ are random variables that represent the measured (observed) state of the system.
The Kalman Filter bnet Fig.21.1 has the following node TPMs, printed in blue:

$$P(x_t|x_{t-1}) = \mathcal{N}(x_t; F_t x_{t-1} + B_t u_t, Q_t) \ , \tag{21.1}$$

where F_t, Q_t, B_t, u_t are given. $P(x_t|x_{t-1})$ becomes $P(x_t)$ for $t = 0$.

$$P(z_t|x_t) = \mathcal{N}(z_t; H_t x_t, R_t) \ , \tag{21.2}$$

where H_t, R_t are given.
Define

$$\underline{Z}_t = (\underline{z}_{t'})_{t' \le t} \ . \tag{21.3}$$

Define $\hat{x}_t$ and P_t by

$$P(x_t|Z_t) = \mathcal{N}(x_t; \hat{x}_t, P_t) \ . \tag{21.4}$$

Problem: Find $\hat{x}_t$ and P_t in terms of

1. current (at time t) given values of F, Q, H, R, B, u

2. current (at time t) observed value of z

3. prior (previous) value (at time $t-1$) of $\hat{x}$ and P.

See Fig.21.2. For that figure,

$$P(\hat{x}_t, P_t | z_t, \hat{x}_{t-1}, P_{t-1}) = \delta(\hat{x}_t, ?)\delta(P_t, ?) \ . \tag{21.5}$$

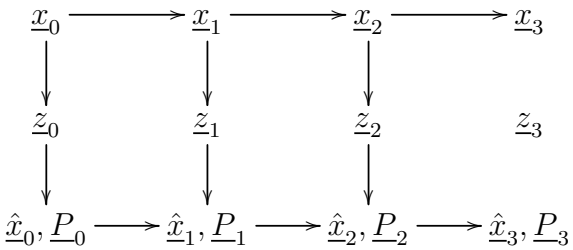


Figure 21.2: Kalman Filter bnet with deterministic nodes for $\hat{x}_t, P_t$.

Solution copied from Wikipedia Ref.[49]:

Define $\eta_{t|t} = \eta_t$ for $\eta = \hat{x}, P$.

- **Predict**

  Predicted (a priori) state estimate

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} + B_t u_t \tag{21.6}$$

  Predicted (a priori) estimate covariance

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^{\mathsf{T}} + Q_t \tag{21.7}$$

- **Update**

  Innovation (or measurement pre-fit residual)

$$\tilde{y}_{t|t-1} = z_t - H_t \hat{x}_{t|t-1} \tag{21.8}$$

  Innovation (or pre-fit residual) covariance

$$S_t = H_t P_{t|t-1} H_t^{\mathsf{T}} + R_t \tag{21.9}$$

Optimal Kalman gain

$$K_t = P_{t|t-1} H_t^{\mathsf{T}} S_t^{-1} \tag{21.10}$$

Updated (a posteriori) state estimate

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \tilde{y}_t \tag{21.11}$$

Updated (a posteriori) estimate covariance

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{21.12}$$

Measurement post-fit residual

$$\tilde{y}_{t|t} = z_t - H_t \hat{x}_{t|t} \tag{21.13}$$

Chapter 22

Linear and Logistic Regression

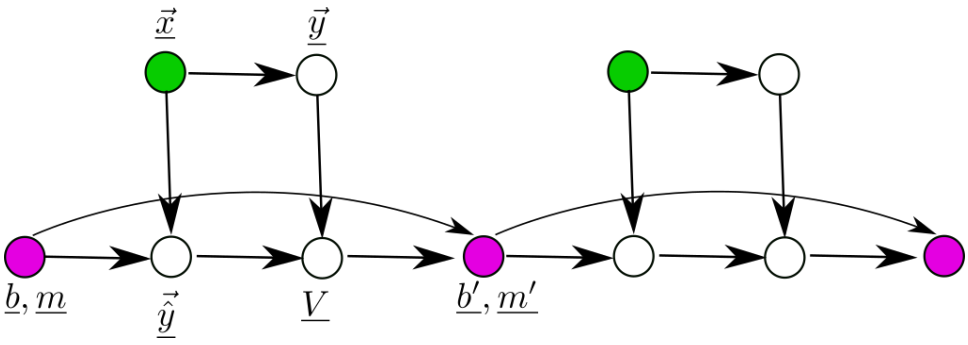


Figure 22.1: Linear Regression

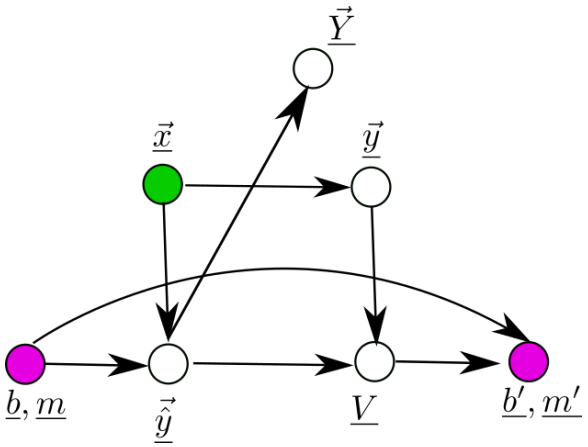


Figure 22.2: B net of Fig.22.1 with new $\vec{Y}$ node.

Estimators $\hat{y}$ for linear and logistic regression.

- **Linear Regression:** $y \in \mathbb{R}$. Note $\hat{y} \in \mathbb{R}$. $(x, \hat{y}(x))$ is the graph of a straight line with

y-intercept b and slope m.

$$\hat{y}(x; b, m) = b + mx \tag{22.1}$$

- **Logistic Regression:** $y \in \{0, 1\}$. Note $\hat{y} \in [0, 1]$. $(x, \hat{y}(x))$ is the graph of a sigmoid. Often in literature, b, m are replaced by β_0, β_1.

$$\hat{y}(x; b, m) = \text{sig}(b + mx) \tag{22.2}$$

Define

$$V(b, m) = \sum_{x,y} P(x, y)|y - \hat{y}(x; b, m)|^2 \ . \tag{22.3}$$

We want to minimize $V(b, m)$ (called a cost or loss function) wrt b and m.

Node TPMs of B net of Fig.22.1 given next in blue.

$$P(b, m) = \text{given} \tag{22.4}$$

The first time it is used, (b, m) is arbitrary. After the first time, it is determined by previous stage.

Let

$$P_{\underline{x},\underline{y}}(x, y) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x = x[\sigma], y = y[\sigma]) \ . \tag{22.5}$$

$$P(\vec{x}) = \prod_{\sigma} P(x[\sigma]) \tag{22.6}$$

$$P(\vec{y}|\vec{x}) = \prod_{\sigma} P(y[\sigma] \mid x[\sigma]) \tag{22.7}$$

$$P(\vec{\hat{y}}|\vec{x}, b, m) = \prod_{\sigma} \delta(\hat{y}[\sigma], \hat{y}(x[\sigma], b, m)) \tag{22.8}$$

$$P(V|\vec{\hat{y}}, \vec{y}) = \delta(V, \frac{1}{nsam(\vec{x})} \sum_{\sigma} |y[\sigma] - \hat{y}[\sigma]|^2) \tag{22.9}$$

Let $\eta_b, \eta_m > 0$. For $x = b, m$, if $x' - x = \Delta x = -\eta \frac{\partial V}{\partial x}$, then $\Delta V \approx \frac{-1}{\eta}(\Delta x)^2 \leq 0$ for $\eta > 0$. This is called "gradient descent".

$$P(b'|V, b) = \delta(b', b - \eta_b \partial_b V) \tag{22.10}$$

$$P(m'|V, m) = \delta(m', m - \eta_m \partial_m V) \tag{22.11}$$

Generalization to x with multiple components(features)

Suppose that for each sample σ, instead of $x[\sigma]$ being a scalar, it has n components called features:

$$x[\sigma] = (x_0[\sigma], x_1[\sigma], x_2[\sigma], \ldots x_{n-1}[\sigma]) \ . \tag{22.12}$$

Slope m is replaced by weights

$$w = (w_0, w_1, w_3, , \ldots, w_{n-1}) \ , \tag{22.13}$$

and the product of 2 scalars $mx[\sigma]$ is replaced by the inner vector product $w^T x[\sigma]$.

Alternative $V(b, m)$ for logistic regression

For logistic regression, since $y[\sigma] \in \{0, 1\}$ and $\hat{y}[\sigma] \in [0, 1]$ are both in the interval $[0, 1]$, they can be interpreted as probabilities. Define probability distributions $p[\sigma](x)$ and $\hat{p}[\sigma](x)$ for $x \in \{0, 1\}$ by

$$p[\sigma](1) = y[\sigma], \quad p[\sigma](0) = 1 - y[\sigma] \tag{22.14}$$

$$\hat{p}[\sigma](1) = \hat{y}[\sigma], \quad \hat{p}[\sigma](0) = 1 - \hat{y}[\sigma] \tag{22.15}$$

Then for logistic regression, the following 2 cost functions $V(b, m)$ can be used as alternatives to the cost function Eq.(22.3) previously given.

$$V(b, m) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} D_{KL}(p[\sigma] \parallel \hat{p}[\sigma]) \tag{22.16}$$

and

$$\begin{aligned}
V(b, m) &= \frac{1}{nsam(\vec{x})} \sum_{\sigma} CE(p[\sigma] \to \hat{p}[\sigma]) && (22.17)\\
&= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \{y[\sigma] \ln \hat{y}[\sigma] + (1 - y[\sigma]) \ln(1 - \hat{y}[\sigma])\} && (22.18)\\
&= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \ln \left\{ \hat{y}[\sigma]^{y[\sigma]} (1 - \hat{y}[\sigma])^{(1 - y[\sigma])} \right\} && (22.19)\\
&= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \ln P(\underline{Y} = y[\sigma] \mid \hat{y} = \hat{y}[\sigma]) && (22.20)\\
&= -\sum_{x,y} P(x, y) \ln P(\underline{Y} = y | \hat{y} = \hat{y}(x, b, m)) && (22.21)
\end{aligned}$$

Above, we used

$$P(\underline{Y} = Y | \hat{y}) = \hat{y}^Y [1 - \hat{y}]^{1 - Y} \tag{22.22}$$

for $Y \in S_{\underline{Y}} = \{0, 1\}$. (Bernoulli distribution).

There is no node corresponding to $\underline{Y}$ in the B net of Fig.22.1. Fig.22.2 shows a new B net that has a new node called $\underline{\vec{Y}}$ compared to the B net of Fig.22.1. One defines the TPMs for all nodes of Fig.22.2 except $\underline{\vec{Y}}$ and $\underline{V}$ the same as for Fig.22.1. For $\underline{\vec{Y}}$ and $\underline{V}$, one defines

$$P(Y[\sigma] \mid \vec{\hat{y}}) = P(\underline{Y} = Y[\sigma] \mid \hat{y}[\sigma]) \tag{22.23}$$

$$P(V|\vec{Y}, \vec{y}) = \delta(V, \frac{-1}{nsam(\vec{x})} \ln \mathcal{L}) \ , \tag{22.24}$$

where $\mathcal{L} = \prod_\sigma P(\underline{Y} = y[\sigma] \mid \hat{y}[\sigma])$=likelihood.

Chapter 23

Linear Deterministic Bnets with External Noise

In this chapter, we will consider bnets which were referred to, prior to the invention of bnets, as: Sewall Wright's **Path Analysis (PA)** and **linear Structural Equations Models (SEM)**. Judea Pearl in his books calls them **linear Structural Causal Models (SCM)**, because they are very convenient for doing causal analysis. We will refer to them as linear Deterministic with External Noise (LDEN) diagrams. This chapter is devoted to LDEN diagrams, except that we will say a few words about non-linear DEN diagrams at the end.

A **DEN diagram** is a special kind of bnet. To build a DEN diagram, start with a deterministic bnet G. The deterministic nodes of G are called the **endogenous (internal) variables**. Now make a bigger bnet $\overline{G}$ called a DEN diagram by adding to each node $\underline{a}$ of G a non-deterministic root node $\underline{u}_{\underline{a}}$ pointing into $\underline{a}$ only. The nodes $\underline{u}_{\underline{a}}$ are called the **exogenous (external) variables**. The exogenous variables make their children noisy. They are assumed to be unobserved and their TPMs are prior probability distributions. Since they are root nodes, they are mutually independent. When we draw a DEN diagram, we will never draw the exogenous nodes, leaving them implicit.

A **linear DEN diagram (LDEN)** is a DEN diagram whose deterministic nodes have a TPM that is a linear function of the states of the parent nodes.

Example of LDEN diagram:

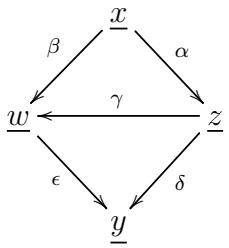


Figure 23.1: Example of a LDEN diagram wherein $\underline{x}$ splits into two nodes $\underline{z}$ and $\underline{w}$, then merges into node $\underline{y}$. There is also an arrow $\underline{z} \rightarrow \underline{w}$. Exogenous nodes are not shown. The Greek letters represent real numbers.

The TPMs, printed in blue, for the nodes of the LDEN diagram Fig.23.1, are as follows.

$$P(y|w,z,u_{\underline{y}}) = \mathbb{1}(y = \epsilon w + \delta z + u_{\underline{y}}) \tag{23.1}$$

$$P(w|x,z,u_{\underline{w}}) = \mathbb{1}(w = \beta x + \gamma z + u_{\underline{w}}) \tag{23.2}$$

$$P(z|x,u_{\underline{z}}) = \mathbb{1}(z = \alpha x + u_{\underline{z}}) \tag{23.3}$$

$$P(x|u_{\underline{x}}) = \mathbb{1}(x = u_{\underline{x}}) \tag{23.4}$$

Hence,

$$\begin{aligned} y &= \epsilon w + \delta z + u_{\underline{y}} && (23.5)\\ &= \epsilon(\beta x + \gamma z + u_{\underline{w}}) + \delta z + u_{\underline{y}} && (23.6)\\ &= (\epsilon\gamma + \delta)z + \epsilon\beta x + \epsilon u_{\underline{w}} + u_{\underline{y}} && (23.7)\\ &= (\epsilon\gamma + \delta)z + \epsilon\beta u_{\underline{x}} + \epsilon u_{\underline{w}} + u_{\underline{y}} \, . && (23.8) \end{aligned}$$

Therefore

$$\left(\frac{\partial y}{\partial z}\right)_{u_{\cdot}-u_{\underline{z}}} = \epsilon\gamma + \delta \, , \tag{23.9}$$

where the partial derivative holds fixed all exogenous variables except $u_{\underline{z}}$. Note that this partial derivative is a sum of terms, and that each of those terms represents a different directed path from $\underline{z}$ to $\underline{y}(\underline{z})$. This is a general property of LDEN diagrams.

Fully Connected LDEN diagrams

The bnets that will be considered in this section will all be fully connected. Fully connected bnets are defined in Chapter 0.2. This section uses the notation $\langle \underline{x}, \underline{y} \rangle$ for the covariance of any two random variables $\underline{x}, \underline{y}$. This $\langle \underline{x}, \underline{y} \rangle$ notation is defined in the Notational Conventions Chapter 0.3.

Consider a LDEN diagram with deterministic nodes $\underline{x}_{\cdot} = (\underline{x}_k)_{k=0,1,\ldots nx-1}$ and corresponding exogenous nodes $\underline{u}_{\cdot} = (\underline{u}_k)_{k=0,1,\ldots nx-1}$. Assume $\langle \underline{u}_i, \underline{u}_j \rangle = 0$ if $i \neq j$. The strength of each connection $\underline{x}_i \to \underline{x}_j$ of the LDEN diagram is measured by a **structural coefficient** $\alpha_{j|i} \in \mathbb{R}$. Some of the $\alpha_{j|i}$ may be zero, in which case the corresponding arrow $i \to j$ would not be drawn.

Fully connected LDEN diagram with $nx = 2$

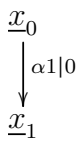


Figure 23.2: Fully connected LDEN diagram with two $\underline{x}_j$ nodes (exogenous nodes $\underline{u}_j$ not shown).

Consider the LDEN diagram of Fig.23.2. This diagram represents the following **structural equations**:

$$\underline{x}_0 = \underline{u}_0 \tag{23.10a}$$
$$\underline{x}_1 = \alpha_{1|0}\underline{x}_0 + \underline{u}_1 \ . \tag{23.10b}$$

Eqs.23.10 constitute a system of 2 linear equations in 2 unknowns (the $\underline{x}$'s) so we can solve for the $\underline{x}$'s in terms of the α's and $\underline{u}$'s.

Note also that

$$\langle \underline{x}_1, \underline{x}_0 \rangle = \alpha_{1|0} \langle \underline{x}_0, \underline{x}_0 \rangle \ . \tag{23.11}$$

Thus, $\alpha_{1|0}$ can be estimated from the covariances $\langle \underline{x}_i, \underline{x}_j \rangle$.

Fully connected LDEN diagram with $nx = 3$

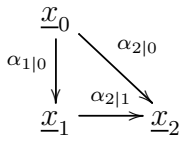


Figure 23.3: Fully connected LDEN diagram with three $\underline{x}_j$ nodes (exogenous nodes $\underline{u}_j$ not shown).

Consider the LDEN diagram of Fig.23.3. This diagram represents the following **structural equations**:

$$\underline{x}_0 = \underline{u}_0 \tag{23.12a}$$
$$\underline{x}_1 = \alpha_{1|0}\underline{x}_0 + \underline{u}_1 \tag{23.12b}$$
$$\underline{x}_2 = \alpha_{2|0}\underline{x}_0 + \alpha_{2|1}\underline{x}_1 + \underline{u}_2 \ . \tag{23.12c}$$

Eqs.23.12 constitute a system of 3 linear equations in 3 unknowns (the $\underline{x}$'s) so we can solve for the $\underline{x}$'s in terms of the α's and $\underline{u}$'s.

Note also that

$$\langle \underline{x}_1, \underline{x}_0 \rangle = \alpha_{1|0} \langle \underline{x}_0, \underline{x}_0 \rangle \tag{23.13a}$$
$$\langle \underline{x}_2, \underline{x}_0 \rangle = \alpha_{2|0} \langle \underline{x}_0, \underline{x}_0 \rangle + \alpha_{2|1} \langle \underline{x}_1, \underline{x}_0 \rangle \tag{23.13b}$$
$$\langle \underline{x}_2, \underline{x}_1 \rangle = \alpha_{2|0} \langle \underline{x}_0, \underline{x}_1 \rangle + \alpha_{2|1} \langle \underline{x}_1, \underline{x}_1 \rangle \tag{23.13c}$$

Eqs.23.13 constitute a system of 3 linear equations in 3 unknowns (the α's) so we can solve solve for the α's in terms of covariances $\langle \underline{x}_i, \underline{x}_j \rangle$. This gives an estimate for the α's.

Fully connected LDEN diagram with arbitrary nx.

Let $\underline{x}. = (x_i)_{i=0,1,\ldots,nx-1}$ and $\underline{x}_{<i} = (x_k)_{k=0,1,\ldots,i-1}$. Consider a fully connected LDEN diagram with deterministic nodes labeled $\underline{x}_i$. The $\underline{x}_i$ labels are assumed to be in **topological order** (i.e., the parents of node $\underline{x}_i$ are $\underline{x}_{<i}$). Let the TPMs, printed in blue, for the nodes $\underline{x}.$ of the LDEN diagram, be

$$P(x_i|x_{<i}, u_i) = \mathbb{1}(x_i = \sum_{k<i} \alpha_{i|k} x_k + u_i) \ , \tag{23.14}$$

for some parameters $\alpha_{i|k} \in \mathbb{R}$. The exogenous nodes $\underline{u}.$ are assumed to be independent so

$$P(u.) = \prod_i P(u_i) \tag{23.15}$$

and

$$\langle \underline{u}_i, \underline{u}_j \rangle = 0 \text{ if } i \neq j \ . \tag{23.16}$$

Note that

$$\begin{aligned} P(x.) &= \sum_{u.} P(u.) \prod_i P(x_i|x_{<i}, u_i) &(23.17)\\ &= E_{\underline{u}.}[\prod_i P(x_i|x_{<i}, u_i)] \ . &(23.18) \end{aligned}$$

In terms of random variables, this system is described by the following **structural equations**:

$$\underline{x}_i = \sum_{k<i} \alpha_{i|k} \underline{x}_k + \underline{u}_i \ . \tag{23.19}$$

The structural equations can be written in matrix form as follows. Define a strictly lower triangular matrix A with the connection strengths $\alpha_{i|k} \in \mathbb{R}$ as entries. For example, for $nx = 4$,

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \alpha_{1|0} & 0 & 0 & 0 \\ \alpha_{2|0} & \alpha_{2|1} & 0 & 0 \\ \alpha_{3|0} & \alpha_{3|1} & \alpha_{3|2} & 0 \end{bmatrix} \ . \tag{23.20}$$

If we now represent the multinodes $\underline{x}.$ and $\underline{u}.$ as column vectors $\underline{x}$ and $\underline{u}$, we get

$$\underline{x} = A\underline{x} + \underline{u} \ . \tag{23.21}$$

Note that

$$\underline{x} = (1 - A)^{-1} \underline{u} \ . \tag{23.22}$$

Therefore,

$$\underline{x}_i = f_i(\underline{u}_{\leq i}) \ . \tag{23.23}$$

Therefore, if $i > j$,

$$\langle \underline{u}_i, \underline{x}_j \rangle = \langle \underline{u}_i, f_j(\underline{u}_{\leq j}) \rangle = 0 \ . \tag{23.24}$$

Thus, if $i > j$,

$$\begin{aligned} \langle \underline{x}_i, \underline{x}_j \rangle &= \sum_{k<i} \alpha_{i|k} \langle \underline{x}_k, \underline{x}_j \rangle + \langle \underline{u}_i, \underline{x}_j \rangle & (23.25) \\ &= \sum_{k<i} \alpha_{i|k} \langle \underline{x}_k, \underline{x}_j \rangle \ . & (23.26) \end{aligned}$$

In matrix notation, Eq.(23.26) becomes

$$\langle \underline{x}, \underline{x}^T \rangle_L = A[\langle \underline{x}, \underline{x}^T \rangle_L + \langle \underline{x}, \underline{x}^T \rangle_D] \tag{23.27}$$

where we are using $\langle \underline{x}, \underline{x}^T \rangle_{i,j} = \langle \underline{x}_i, \underline{x}_j \rangle$ and denoting the strictly lower triangular part and diagonal part of a matrix M by M_L and M_D. Thus,

$$A = \langle \underline{x}, \underline{x}^T \rangle_L [\langle \underline{x}, \underline{x}^T \rangle_L + \langle \underline{x}, \underline{x}^T \rangle_D]^{-1} \ . \tag{23.28}$$

This gives an estimate for the α's in terms of the covariances $\langle \underline{x}_i, \underline{x}_j \rangle$.

Non-linear DEN diagrams

This chapter is dedicated to linear DEN diagrams. This implicitly assumes that the deterministic nodes $\underline{x}.$ of the DEN diagram have an interval of real values as their possible states. A trivial but very useful generalization of linear DEN diagrams is to replace Eq.(23.14) for the TPMs of the deterministic nodes of the diagram by

$$P(x_i | x_{<i}, u_i) = \mathbb{1}(x_i = f_i(x_{<i}, u_i)) \ , \tag{23.29}$$

with structural equations

$$\underline{x}_i = f_i(\underline{x}_{<i}, \underline{u}_i) \ , \tag{23.30}$$

for $i = 0, 1, \ldots, nx-1$. Here the f_i are possibly non-linear functions that depend the states $x_{<i}$ and u_i of nodes $\underline{x}_{<i}$ and $\underline{u}_i$. If a node $\underline{x}_i$ has no arrows entering it (i.e., is a root node), then

$$P(x_i | x_{<i}, u_i) = P(x_i) = \delta(x_i, a) \tag{23.31}$$

and

$$\underline{x}_i = a \tag{23.32}$$

for some $a \in S_{\underline{x}_i}$.

With this generalization, we can make any $f_i()$ represent a continuous probability distribution such as a Gaussian, or a discrete-valued Boolean function such as an OR gate.

Eqs.(23.29) and (23.30) are the TPMs and structural equations for a fully connected, nonlinear DEN diagram. For a non-fully connected diagram,

- replace the multinode $x_{<i}$ by a subset of itself, in Eqs.(23.29) and (23.30) , and
- delete the corresponding arrows from the graph.

Chapter 24

Markov Blankets

This chapter is based on the Wikipedia article, Ref.[51]. Markov blankets and Markov boundaries of bnets were apparently invented by Judea Pearl. His 1988 book Ref.[18], instead of a research paper, is usually given as the original reference.

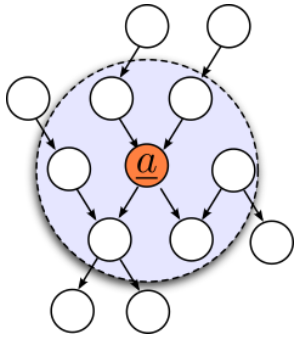


Figure 24.1: In a bnet, the minimal Markov blanket, aka Markov boundary, of node $\underline{a}$.

We will treat vectors of random variables as if they were sets when using the $\in$, $\subset$ and $-$ operations. For example, if $\underline{x} = (\underline{x}_0, \underline{x}_1, \underline{x}_2, \underline{x}_3)$ and $\underline{b} = (\underline{x}_1, \underline{x}_2)$, then $\underline{x}_1 \in \underline{b} \subset \underline{x}$ and $\underline{x} - \underline{b} = (\underline{x}_0, \underline{x}_3)$.

Below, $H(\underline{a} : \underline{b}|\underline{c})$ denotes the conditional mutual information of random variables $\underline{a}$ and $\underline{b}$ conditioned on random variable $\underline{c}$. $H(\underline{a} : \underline{b}|\underline{c})$ is used in Shannon Information Theory, where it is defined by

$$H(\underline{a} : \underline{b}|\underline{c}) = \sum_{a,b,c} P(a,b,c) \ln \frac{P(a,b|c)}{P(a|c)P(b|c)} \ . \tag{24.1}$$

$H(\underline{a} : \underline{b}|\underline{c}) = 0$ iff $\underline{a}$ and $\underline{b}$ are independent (uncorrelated) when $\underline{c}$ is held fixed.

Suppose $\underline{a} \in \underline{X}$, $\underline{B} \subset \underline{X}$, but $\underline{a} \notin \underline{B}$. Then $\underline{B}$ is a Markov blanket of $\underline{a}$ if

$$H(\underline{a} : \underline{X} - \underline{a}|\underline{B}) = 0 \ . \tag{24.2}$$

In other words, one may assume that $\underline{a}$ depends on $\underline{B}$ only, and is independent of all random variables in $\underline{X} - (\underline{a} \cup \underline{B})$.

The minimal Markov blanket is called the Markov boundary.
In a bnet, the Markov boundary of a node $\underline{a}$, contains:

1. the parents of $\underline{a}$,
2. the children of $\underline{a}$,
3. the parents, other than $\underline{a}$, of the children of $\underline{a}$.

This is illustrated in Fig.24.1.

Chapter 25

Markov Chain Monte Carlo (MCMC)

Monte Carlo methods are methods for using random number generation to sample probability distributions. The subject of Monte Carlo methods has many branches, as you can see from its Wikipedia category list, Ref.[54]. MCMC (Markov Chain Monte Carlo) is just one of those branches, albeit a major one. Metropolis-Hastings (MH) sampling is a very important MCMC method. Gibbs sampling is a special case of MH sampling. This chapter covers both, MH and Gibbs sampling. It also covers a few other types of sampling.

Throughout this chapter, we use $P_{\underline{x}} : S_{\underline{x}} \to [0, 1]$ to denote the target probability distribution that we wish to obtain samples from.

Inverse Cumulative Sampling

For more info about this topic and some original references, see Ref.[46].

This is one of the simplest methods for obtaining samples from a probability distribution $P_{\underline{x}}$, but it requires knowledge of the inverse cumulative distribution of $P_{\underline{x}}$, which is often not available.

The **cumulative distribution** function is defined by:

$$CUM_{\underline{x}}(x) = P(\underline{x} < x) = \int_{x' < x} dx' \; P_{\underline{x}}(x') \; . \tag{25.1}$$

Note that

$$P_{\underline{x}}(x) = \frac{d}{dx} CUM_{\underline{x}}(x) \; . \tag{25.2}$$

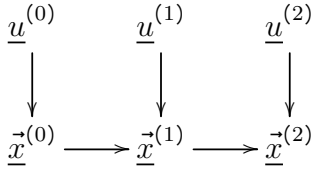


Figure 25.1: bnet for Inverse Cumulative Sampling

For $t = 0, 1, \ldots, T-1$, let

$\underline{u}^{(t)} \in [0,1]$= random variable, uniformly distributed over $[0,1]$.

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all σ. Vector of samples collected up to time t.

The TPMs, printed in blue, for the nodes of bnet Fig.25.1, are:

$$P(u^{(t)}) = 1 \tag{25.3}$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, u^{(t)}) = \delta(\quad \vec{x}^{(t)}, [\vec{x}^{(t-1)}, CUM_{\underline{x}}^{-1}(u^{(t)})]\quad) \tag{25.4}$$

Motivation

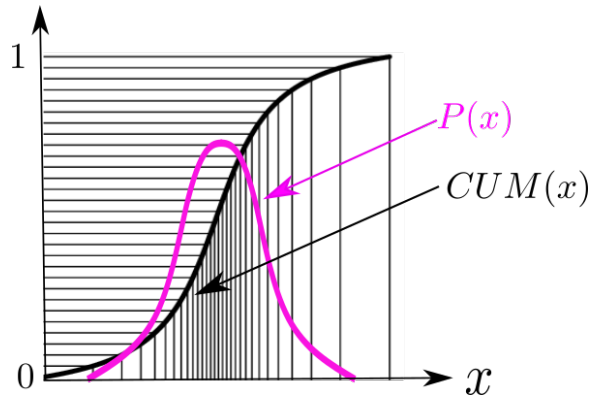


Figure 25.2: Motivation for Inverse Cumulative Sampling.

See Fig.25.2.

Note that if $\underline{u}$ is uniformly distributed over the interval $[0,1]$ and $a \in [0,1]$, then

$$P(\underline{u} < a) = a\ . \tag{25.5}$$

Thus

$$\begin{aligned} P(CUM_{\underline{x}}^{-1}(\underline{u}) < x) &= P(\underline{u} < CUM_{\underline{x}}(x)) && (25.6)\\ &= CUM_{\underline{x}}(x)\ . && (25.7) \end{aligned}$$

Therefore,

$$dP(CUM_{\underline{x}}^{-1}(\underline{u}) < x) = P_{\underline{x}}(x)dx\ . \tag{25.8}$$

Rejection Sampling

For more info about this topic and some original references, see Ref.[60].

This method samples from a "candidates" probability distribution $P_{\underline{c}} : S_{\underline{x}} \to [0,1]$, in cases where sampling directly from the target probability distribution $P_{\underline{x}} : S_{\underline{x}} \to [0,1]$ is not possible.

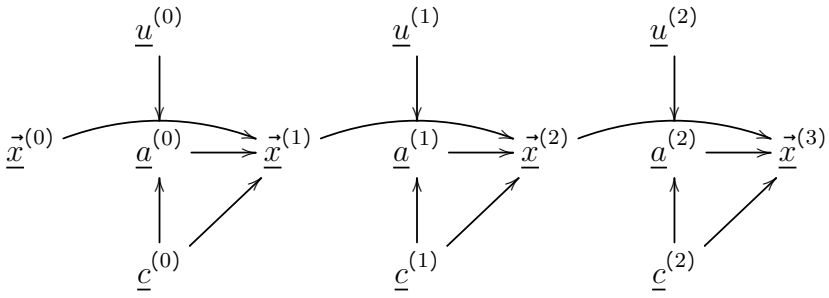


Figure 25.3: bnet for Rejection Sampling

For $t = 0, 1, \ldots, T-1$, let

$\underline{u}^{(t)} \in [0,1]$= random variable, uniformly distributed over $[0,1]$.

$\underline{a}^{(t)} \in \{0,1\}$= accept candidate? (no=0, yes=1)

$\underline{c}^{(t)} \in S_{\underline{x}}$= sample that is a candidate for being accepted

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all σ. Vector of samples collected up to time t.

This algorithm requires a priori definition of a candidate probability distribution $P_{\underline{c}} : S_{\underline{x}} \to \mathbb{R}$ such that

$$P_{\underline{x}}(x) < \beta P_{\underline{c}}(x) \tag{25.9}$$

for all $x \in S_{\underline{x}}$, for some $\beta \in \mathbb{R}$.

The TPMs, printed in blue, for the nodes of bnet Fig.25.3, are:

$$P(\underline{u}^{(t)} = u) = 1 \tag{25.10}$$

$$P(\underline{c}^{(t)} = c) = P_{\underline{c}}(c) \tag{25.11}$$

$$P(\underline{a}^{(t)} = a | \underline{c}^{(t)} = c, \underline{u}^{(t)} = u) = \begin{cases} \delta(a,0) & \text{if } u\beta P_{\underline{c}}(c) \geq P_{\underline{x}}(c) \\ \delta(a,1) & \text{if } u\beta P_{\underline{c}}(c) < P_{\underline{x}}(c) \end{cases} \tag{25.12}$$

$$P(\vec{\underline{x}}^{(t)} | \vec{\underline{x}}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{\underline{x}}^{(t)}, \vec{\underline{x}}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{\underline{x}}^{(t)}, [\vec{\underline{x}}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \tag{25.13}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\vec{x}^{(0)})$ which must be specified a priori.

Motivation

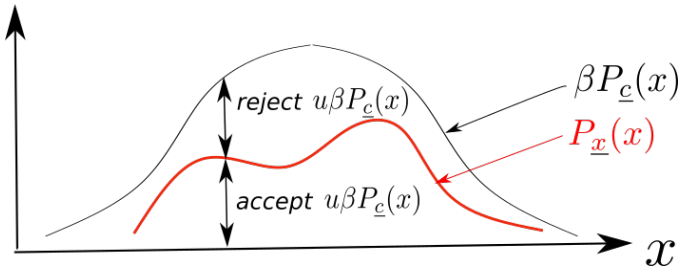


Figure 25.4: Motivation for Rejection Sampling.

See Fig.25.4.

Metropolis-Hastings Sampling

For more info about this topic and some original references, see Refs.[1] and [52].

An advantage of this method is that it can sample unnormalized probability distributions $(constant)P_{\underline{x}}$ because it only uses ratios of $P_{\underline{x}}$ at two different points. Another advantage of this method is that it scales much better than other sampling methods as the number of dimensions of the sampled variable $\underline{x}$ increases.

This method produces samples that take a finite amount of time to reach steady state. The samples are also theoretically correlated instead of being i.i.d. as one desires. To mitigate for the steady state problem, one discards an initial set of samples (the "burn-in" period). To mitigate for the correlation problem, one calculates the autocorrelation between the samples and keeps only samples separated by a time interval after which the samples cease to be autocorrelated to a good approximation.

For $t = 0, 1, \ldots, T-1$, let

$\underline{u}^{(t)} \in [0, 1]$= random variable, uniformly distributed over $[0, 1]$.

$\underline{a}^{(t)} \in \{0, 1\}$= accept candidate? (no=0, yes=1)

$\underline{c}^{(t)} \in S_{\underline{x}}$= sample that is a candidate for being accepted

$\underline{m}^{(t)} \in S_{\underline{x}}$= memory of last accepted sample

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\ldots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all σ. Vector of samples collected up to time t.

A **proposal TPM** $P_{\underline{c}|\underline{x}} : S_{\underline{x}}^2 \rightarrow [0, 1]$ must be specified a priori for this algorithm.

The TPMs, printed in blue, for the nodes of bnet Fig.25.5, are:

$$P(u^{(t)} = u) = 1 \tag{25.14}$$

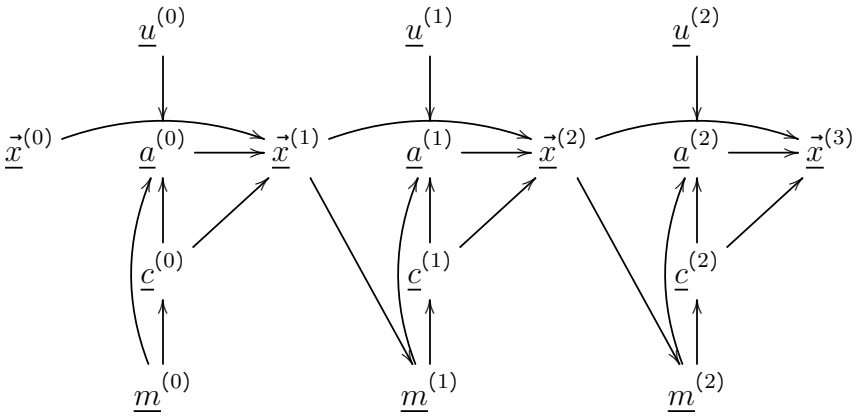


Figure 25.5: bnet for Metropolis-Hastings Sampling

$$P(\underline{c}^{(t)} = c|\underline{m}^{(t)} = m) = P_{\underline{c}|\underline{x}}(c|m) \tag{25.15}$$

$$P(\underline{a}^{(t)} = a|\underline{c}^{(t)} = c, \underline{u}^{(t)} = u, \underline{m}^{(t)} = m) = \begin{cases} \delta(a,0) & \text{if } u \geq \alpha(c|m) \\ \delta(a,1) & \text{if } u < \alpha(c|m) \end{cases} \tag{25.16}$$

where the **acceptance probability** α is defined as

$$\alpha(c|m) = \min\left(1, \frac{P_{\underline{c}|\underline{x}}(m|c)P_{\underline{x}}(c)}{P_{\underline{c}|\underline{x}}(c|m)P_{\underline{x}}(m)}\right) . \tag{25.17}$$

Note that if the proposal distribution is symmetric, then

$$\alpha(c|m) = \min\left(1, \frac{P_{\underline{x}}(c)}{P_{\underline{x}}(m)}\right) . \tag{25.18}$$

$$P(\vec{\underline{x}}^{(t)}|\vec{\underline{x}}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{\underline{x}}^{(t)}, \vec{\underline{x}}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{\underline{x}}^{(t)}, [\vec{\underline{x}}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \tag{25.19}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\vec{\underline{x}}^{(0)})$ which must be specified a priori.

$$P(\underline{m}^{(t)} = m|\vec{\underline{x}}^{(t)}) = \delta(m, \text{last component of } \vec{\underline{x}}^{(t)}) . \tag{25.20}$$

This last equation is only defined for $t > 0$. For $t = 0$, the left hand side reduces to $P(\underline{m}^{(0)} = m)$ which must be specified a priori.

Motivation

See Fig.25.6.

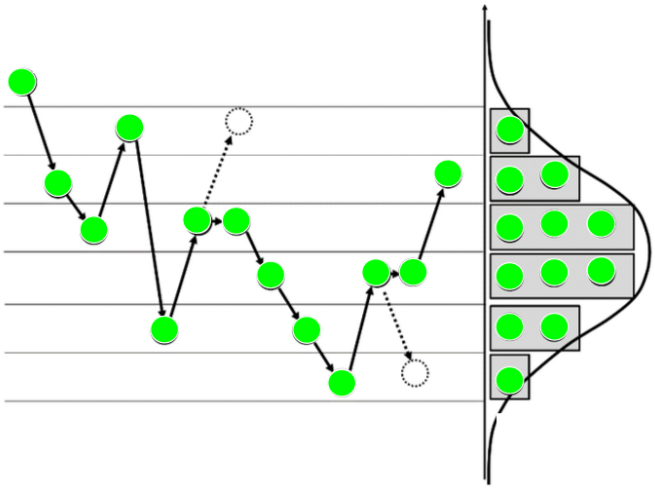


Figure 25.6: Motivation for Metropolis-Hastings Sampling.

Consider a time homogeneous (its TPM is the same for all times) Markov chain with TPM $P(x'|x) = [T]_{x',x}$. Its **stationary distribution**, if it exists, is defined as

$$\pi = \lim_{n\to\infty} T^n \pi_0 \ . \tag{25.21}$$

Suppose the prob distribution $P_{\underline{x}}(x)$ that we wish to sample from satisfies

$$P_{\underline{x}}(x) = \pi(x) \ . \tag{25.22}$$

Reversibility (detailed balance): For all $x, x' \in S_{\underline{x}}$,

$$P(x'|x)\pi(x) = P(x|x')\pi(x') \ . \tag{25.23}$$

Detailed balance is a sufficient (although not necessary) condition for a unique stationary prob distribution π to exist.[1]

Let

$$P(x'|x) = P(\underline{a} = 1|x', x)P_{\underline{c}|\underline{x}}(x'|x) + \delta(x, x')P(\underline{a} = 0|x) \ , \tag{25.24}$$

where

$$P(\underline{a} = 0|x) = \sum_{x'} P(\underline{a} = 0|x', x)P_{\underline{c}|\underline{x}}(x'|x) \ . \tag{25.25}$$

Claim 18 *If*

$$P(\underline{a} = 1|x', x) = \alpha(x'|x) \ , \tag{25.26}$$

[1] As explained lucidly in Ref.[1], besides detailed balance, 2 other properties must also be satisfied by the Markov chain, irreducibiity and aperiodicity. However, because of how it is constructed, the Metropolis-Hastings algorithm automatically produces a Markov chain that has those 2 properties.

then detailed balance is satisfied.

proof: Assume $x \neq x'$.

$$\begin{aligned} P(x'|x)P(x) &= P(\underline{a}=1|x',x)P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) && (25.27)\\ &= \min\left(1, \frac{P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')}{P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x)}\right) P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) && (25.28)\\ &= \min\Big(P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x), P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')\Big) && (25.29)\\ &= P(x|x')P(x') && (25.30) \end{aligned}$$

QED

Gibbs Sampling

For more info about this topic and some original references, see Ref.[43].

Gibbs sampling is a special case of Metropolis-Hastings sampling. Gibbs sampling is ideally suited for application to a bnet, because it is stated in terms of the conditional prob distributions of N random variables, and conditional prob distributions are part of the definition of a bnet.

Consider a bnet with nodes $\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{N-1}$

Identify the random variable $\underline{x} = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{N-1})$ with the random variable $\underline{x}$ used in Metropolis-Hastings sampling. For Gibbs sampling, we use the following proposal distribution:

$$P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j \mid [m_i]_{i\neq j}) \; . \tag{25.31}$$

Eq.(25.31) can be simplified using Markov Blankets (see Chapter 24) to the following:

$$P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j \mid [m_i : \forall i \ni \underline{x}_i \in MB(\underline{x}_j)]) \; , \tag{25.32}$$

where, for any node $\underline{a}$, we denote its Markov blanket by $MB(\underline{a})$.

An alternative proposal distribution that leads to much faster convergence is as follows. The idea is to make the components $c_j^{(t)}$ of candidate sample $c^{(t)}$ depend on the previous components $(c_i^{(t)})_{i<j}$. See the bnet Fig.25.7. The TPM for the nodes of that bnet are

$$P(\underline{c}_j^{(t)} = c_j \mid (\underline{c}_i^{(t)})_{i<j} = (c_i)_{i<j}, \underline{m}^{(t-1)} = m) = P(c_j|(c_i)_{i<j}, (m_i)_{i>j}) \tag{25.33}$$

for $j = 0, 1, \ldots, N-1$. This implies

$$P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c|\underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j|(c_i)_{i<j}, (m_i)_{i>j}) \; . \tag{25.34}$$

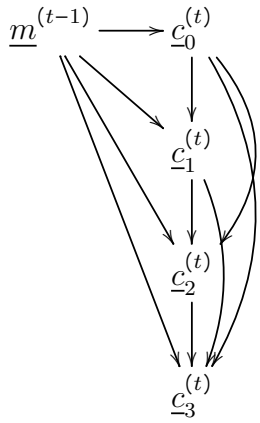


Figure 25.7: In Gibbs sampling, the proposal distribution $P_{\underline{c}|\underline{x}}$ can be defined by making the components $c_j^{(t)}$ of candidate sample $c^{(t)}$ depend on the previous components $(c_i^{(t)})_{i<j}$.

As before, we can condition only on the Markov blanket of each node $\underline{x}_j$.

$$P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c|\underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j|(c_i)_{i<j}, (m_i)_{i>j}, \text{ use only } c_i \text{ and } m_i \ni \underline{x}_i \in MB(\underline{x}_j)) \ . \tag{25.35}$$

Importance Sampling

For more info about this topic and some original references, see Ref.[45].

Suppose random variables $\underline{x}[\sigma] \in S_{\underline{x}}$ for $\sigma = 0, 1, \ldots, nsam - 1$ are i.i.d. with probability distribution $P_{\underline{x}}$. Then

$$E_{\underline{x}}[f(x)] \approx \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} f(x[\sigma]) \tag{25.36}$$

for any $f : S_{\underline{x}} \to \mathbb{R}$. Sometimes, instead of using i.i.d. samples $\underline{x}[\sigma] \in S_{\underline{x}}$ where $\underline{x}[\sigma] \sim P_{\underline{x}}$, we wish to use i.i.d. samples $\underline{y}[\sigma] \in S_{\underline{x}}$ where $\underline{y}[\sigma] \sim P_{\underline{y}}$.

$$E_{\underline{x}}[f(\underline{x})] = \sum_x P_{\underline{x}}(x) f(x) \tag{25.37}$$

$$= \sum_x P_{\underline{y}}(x)) \frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} f(x) \tag{25.38}$$

$$= E_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)] \tag{25.39}$$

Sampling from $P_{\underline{y}}(y)$ instead of $P_{\underline{x}}(x)$ might reduce (or increase) variance for a particular $f : S_{\underline{x}} \rightarrow \mathbb{R}$.

$$Var_{\underline{x}}[f(x)] = E_{\underline{x}}[(f(x))^2] - (E_{\underline{x}}[f(x)])^2 \tag{25.40}$$

$$\begin{aligned} Var_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)] &= E_{\underline{y}}[(\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y))^2] - (E_{\underline{y}}[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)])^2 & (25.41) \\ &= E_{\underline{x}}[\frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} (f(x))^2] - (E_{\underline{x}}[f(x)])^2 & (25.42) \end{aligned}$$

Chapter 26

Markov Chains

A Markov Chain is simply a bnet with the graph structure of a chain. For example, Fig.26.1 shows a chain with $n = 4$ nodes.

$$\underline{x}_0 \longrightarrow \underline{x}_1 \longrightarrow \underline{x}_2 \longrightarrow \underline{x}_3$$

Figure 26.1: Markov chain with $n = 4$ nodes.

Because of its graph structure, the TPM of each node only depends on the state of the previous node:

$$P(x_t|(x_a)_{a\neq t}) = P(x_t|x_{t-1}) \ , \tag{26.1}$$

where $(x_a)_{a\neq t}$ are all the nodes except x_t itself and $t = 1, 2, \ldots, n-1$.

If there exists a single TPM $P_{\underline{x}_1|\underline{x}_0}$ such that

$$P(x_t|x_{t-1}) = P_{\underline{x}_1|\underline{x}_0}(x_t|x_{t-1}) \tag{26.2}$$

for $t = 1, 2, \ldots, n-1$, then we say that the Markov chain is **time homogeneous**.

Claim 19 *(Data Processing Inequality (DPI))*
Consider a Markov chain $\underline{x}_0 \to \underline{x}_1 \cdots \to \underline{x}_{n-1}$. *Suppose* $0 \le a < m < b \le n-1$. *Then*

$$H(\underline{x}_a : \underline{x}_b) \le \min[H(\underline{x}_a : \underline{x}_m), H(\underline{x}_m : \underline{x}_b)] \tag{26.3}$$

See Ref.[38] for references where the DPI is proven. This inequality confirms our intuitive expectations that the information transmitted (i.e., the mutual information(MI)) from $\underline{a}$ to $\underline{b}$ (or vice versa since MI is symmetric) is smaller or equal to the one transmitted from $\underline{a}$ to $\underline{m}$ or from $\underline{m}$ to $\underline{b}$ because $\underline{a}$ and $\underline{b}$ are "farther apart" and "some info can get lost during transmission through the mediator node $\underline{m}$".

Chapter 27

Message Passing (Belief Propagation)

Belief Propagation was first proposed in 1982 Ref.[17] by Judea Pearl to simplify the exact evaluation of the probability of one node conditioned on other nodes of a bnet (exact inference). It gives exact results for trees and polytrees (i.e. bnets with a single connected component and no loops). For bnets with loops, it gives approximate results (loopy belief propagation), and it has been generalized to the junction tree algorithm (see Chapter 20) which can do exact inference for general bnets with loops. The basic idea behind the junction tree algorithm is to eliminate loops by clustering them into single nodes.

In his book Ref.[18], Pearl explains two types of Message Passing (i.e., distributed computing in a bnet). In Chapter 4, he discusses one type of MP which he calls Belief Propagation (BP) or Belief Updating. In Chapter 5, he introduces a second type of MP which is he calls Belief Revision, but which I prefer to call Explanation Optimization (EO). This chapter will be devoted to BP only.

This chapter is mostly based on chapter 4 of Ref.[18] by Pearl. Refs.[32], and [13] were also helpful in writing this chapter.

Distributed Soldier Counting

Consider a group of soldiers marching single file. Fig.27.1 shows several methods by which a member of the group can obtain a count of the soldiers without breaking the line to do global operations. This can be done in a distributed fashion, with every soldier doing only local operations (i.e., each soldier can only send messages to either the soldier in front or the one in back). Such distributed soldier counting is a rudimentary type of BP. In the next section, we will generalize this BP for soldiers to BP for a Markov chain.

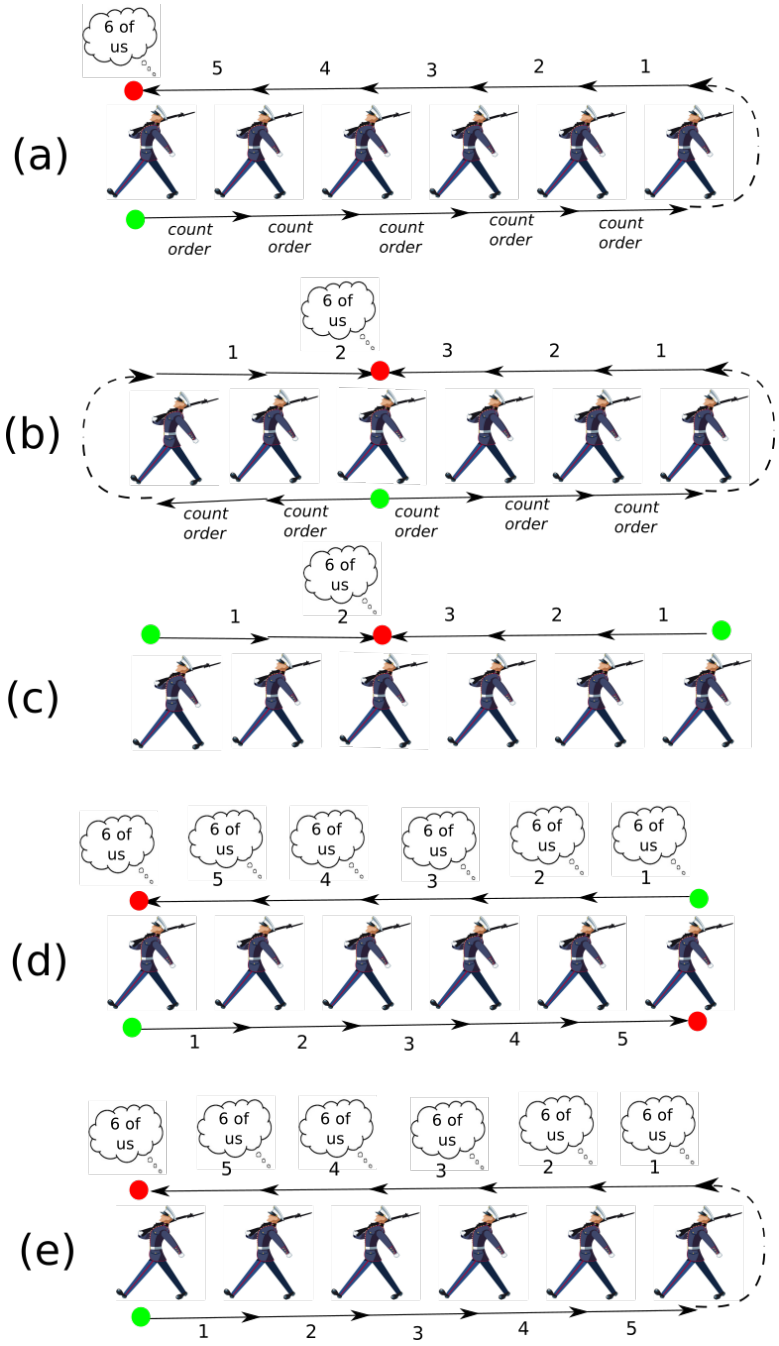


Figure 27.1: Distributed soldier counting (This example comes from Chapter 4 of Ref.[18]). Green dots indicate the beginning and red dots the end of a count. Only first soldier can calculate total count in (a). Only third soldier can calculate total count in (b,c). All soldiers can calculate the total count in (d,e). One starting point in (a,b,e). Two ends as starting points in (c,d).

BP for Markov Chains

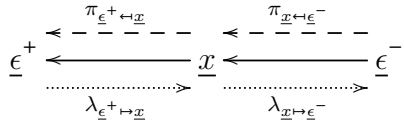


Figure 27.2: 3 node Markov chain $\underline{\epsilon}^+ \leftarrow \underline{x} \leftarrow \underline{\epsilon}^-$. The π messages (probability functions) travel downstream (i.e., in the direction of the graph arrows, towards the future) and are indicated by dashed arrows. The λ messages (likelihood functions) travel upstream (i.e., opposite to direction of the graph arrows, towards the past) and are indicated by dotted arrows. I call the symbols $\mapsto$ and $\mapsto$, a left dart and a right dart. These darts indicate the direction in which the π and λ messages travel. $\underline{\epsilon}^+$ stands for future evidence and $\underline{\epsilon}^-$ for past evidence.

Consider the 3 node Markov chain $\underline{\epsilon}^+ \leftarrow \underline{x} \leftarrow \underline{\epsilon}^-$ shown in Fig.27.2. Define

$$\pi_{\underline{\epsilon}^+ \mapsto \underline{x}}(x) = P(x|\epsilon^-) \text{ (past of } \underline{x}) \tag{27.1}$$

$$\lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}(x) = P(\epsilon^+|x) \text{ (future of } \underline{x}) \tag{27.2}$$

$$\pi_{\underline{x} \mapsto \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^-) = \delta(\epsilon^-, \epsilon_0^-) \text{ (past of } \underline{\epsilon}^-) \tag{27.3}$$

$$\lambda_{\underline{x} \mapsto \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^+|\epsilon^-) \text{ (future of } \underline{\epsilon}^-) \tag{27.4}$$

Furthermore, define the Belief BEL in x to be

$$BEL_{\underline{x}}(x) = P(x|\epsilon) \,, \tag{27.5}$$

where

$$\underline{\epsilon} = \underline{\epsilon}^+ \cup \underline{\epsilon}^- \,. \tag{27.6}$$

It follows that

$$\begin{aligned} BEL_{\underline{x}}(x) &= P(x|\epsilon^+, \epsilon^-) = & (27.7)\\ &= \mathcal{N}(!x) P(\epsilon^+, x, \epsilon^-) & (27.8)\\ &= \mathcal{N}(!x) P(\epsilon^+|x) P(x|\epsilon^-) & (27.9)\\ &= \mathcal{N}(!x) \lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}(x) \pi_{\underline{\epsilon}^+ \mapsto \underline{x}}(x) \,. & (27.10) \end{aligned}$$

Note that Bayes rule would affirm that

$$P(x|\epsilon^+) = \mathcal{N}(!x) \underbrace{P(\epsilon^+|x)}_{\lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}(x)} P(x) \,. \tag{27.11}$$

Thus, Eq.(27.10) is like a 2-sided Janus Bayes rule.

Note that the π messages and λ messages propagate independently of each other, via the TPM $P(x|\epsilon^-)$:

$$\underbrace{\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x)}_{P(x|\epsilon_0^-)} = \sum_{\epsilon^-} P(x|\epsilon^-) \underbrace{\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-)}_{\delta(\epsilon^-, \epsilon_0^-)} \tag{27.12a}$$

$$\underbrace{\lambda_{\underline{x} \mapsto \underline{\epsilon}^-}(\epsilon^-)}_{P(\epsilon^+|\epsilon^-)} = \sum_{x} P(x|\epsilon^-) \underbrace{\lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}(x)}_{P(\epsilon^+|x)} \tag{27.12b}$$

Eqs.(27.12) suggest that we define an edge bnet for the π and λ messages (these messages live in the edges between the nodes $\underline{\epsilon}^+, \underline{x}, \underline{\epsilon}^-$). Such an edge bnet, shown in Fig.27.3, is complementary to bnet for the nodes themselves. We will call it the **BP 2-track bnet** for the bnet Fig.27.2, because it has two "tracks", one for π messages and another for λ ones. The TPMs, shown in blue, for the nodes of bnet Fig.27.3, are as follows:

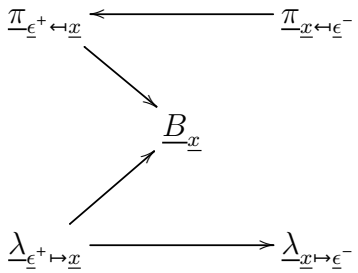


Figure 27.3: BP 2-track bnet for the bnet Fig.27.2.

$$P(\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}) = \prod_{\epsilon^-} \mathbb{1}(\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^-)) \tag{27.13}$$

$$P(\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}} | \pi_{\underline{x} \leftarrow \underline{\epsilon}^-}) = \prod_{x} \mathbb{1}\left(\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x) = \sum_{\epsilon^-} P(x|\epsilon^-) \pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-)\right) \tag{27.14}$$

$$P(B_{\underline{x}} | \pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}, \lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}) = \prod_{x} \mathbb{1}\left(B_{\underline{x}}(x) = BEL_{\underline{x}}(x)\right) \tag{27.15}$$

$$P(\lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}) = \prod_{x} \mathbb{1}\left(\lambda_{\underline{\epsilon}^+ \mapsto \underline{x}}(x) = P(\epsilon_0^+|x)\right) \tag{27.16}$$

$$P(\lambda_{\underline{x}\mapsto\underline{\epsilon}^-}|\lambda_{\underline{\epsilon}^+\mapsto\underline{x}}) = \prod_{\epsilon^-} \mathbb{1}\left(\lambda_{\underline{x}\mapsto\underline{\epsilon}^-}(\epsilon^-) = \sum_x P(x|\epsilon^-)\lambda_{\underline{\epsilon}^+\mapsto\underline{x}}(x)\right) \tag{27.17}$$

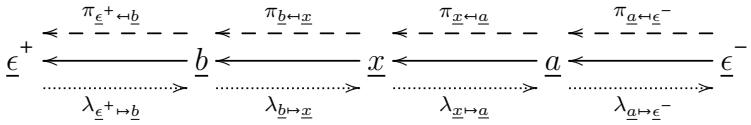


Figure 27.4: 5 node Markov chain

So far in this section, we have considered Markov chains with 3 nodes. Before concluding our discussion of BP for Markov chains, let us consider BP for a slightly longer chain. Let us consider the 5 node Markov chain $\underline{\epsilon}^+ \leftarrow \underline{b} \leftarrow \underline{x} \leftarrow \underline{a} \leftarrow \underline{\epsilon}^-$ shown in Fig.27.4. We have already dealt with the end nodes of a Markov chain in the 3 node Markov chain example above, so in the 5 node case, let us focus on the internal (i.e., not at an end) node $\underline{x}$ and its neighbors $\underline{a}$ and $\underline{b}$. Define

$$\pi_{\underline{b}\mapsto\underline{x}}(x) = P(x|\epsilon^-) \text{ (past of } \underline{x}) \,, \tag{27.18}$$

$$\lambda_{\underline{b}\mapsto\underline{x}}(x) = P(\epsilon^+|x) \text{ (future of } \underline{x}) \,, \tag{27.19}$$

$$\pi_{\underline{x}\mapsto\underline{a}}(a) = P(a|\epsilon^-) \text{ (past of } \underline{a}) \tag{27.20}$$

and

$$\lambda_{\underline{x}\mapsto\underline{a}}(a) = P(\epsilon^+|a) \text{ (future of } \underline{a}) \,. \tag{27.21}$$

Define the Belief BEL in x to be

$$BEL_{\underline{x}}(x) = P(x|\epsilon) \,, \tag{27.22}$$

where

$$\underline{\epsilon} = \underline{\epsilon}^+ \cup \underline{\epsilon}^- \,. \tag{27.23}$$

Then

$$BEL_{\underline{x}}(x) = \mathcal{N}(!x)P(\epsilon^+|x)P(x|\epsilon^-) \tag{27.24}$$

$$= \mathcal{N}(!x)\lambda_{\underline{b}\mapsto\underline{x}}(x)\pi_{\underline{b}\mapsto\underline{x}}(x) \,. \tag{27.25}$$

In analogy with the case of BP for a 3 node Markov chain, we can define the bnet Fig.27.5, which we refer to as the **BP 2-track bnet** for Fig.27.4. The TPMs, printed in blue, for the nodes of bnet Fig.27.5, are as follows:

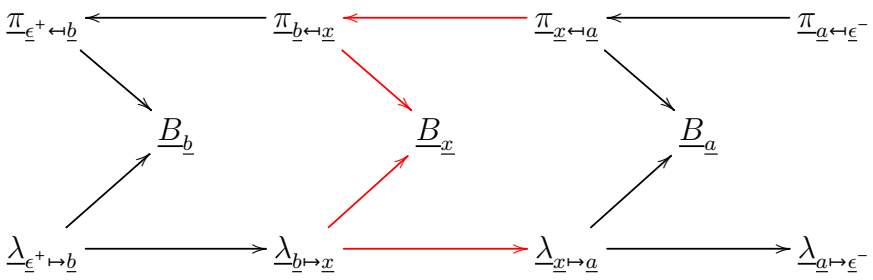


Figure 27.5: BP 2-track bnet for the bnet Fig.27.4.

$$P(\pi_{\underline{b} \mapsto \underline{x}} | \pi_{\underline{x} \mapsto \underline{a}}) = \prod_x \mathbb{1}\left(\pi_{\underline{b} \mapsto \underline{x}}(x) = \sum_a P(x|a)\pi_{\underline{x} \mapsto \underline{a}}(a)\right) \tag{27.26}$$

$$P(B_{\underline{x}} | \pi_{\underline{b} \mapsto \underline{x}}, \lambda_{\underline{b} \mapsto \underline{x}}) = \prod_x \mathbb{1}\left(B_{\underline{x}}(x) = BEL_{\underline{x}}(x)\right) \tag{27.27}$$

$$P(\lambda_{\underline{x} \mapsto \underline{a}} | \lambda_{\underline{b} \mapsto \underline{x}}) = \prod_a \mathbb{1}\left(\lambda_{\underline{x} \mapsto \underline{a}}(a) = \sum_x P(x|a)\lambda_{\underline{b} \mapsto \underline{x}}(x)\right) \tag{27.28}$$

Let us represent the Markov chain of Fig.27.4 by $\underline{x}_{nx-1} \leftarrow \ldots, \underline{x}_2 \leftarrow \underline{x}_1 \leftarrow \underline{x}_0$ where $nx = 5$. For any node $\underline{x}_i$ with parent $\underline{px}_i = \underline{x}_{i-1}$ and child $\underline{cx}_i = \underline{x}_{i+1}$, define the **memory matrix** $\mathcal{M}_{\underline{x}_i}$ for node $\underline{x}_i$ as

$$\mathcal{M}_{\underline{x}_i} = [\mathcal{M}^+_{\underline{x}_i}, \mathcal{M}^-_{\underline{x}_i}], \tag{27.29}$$

where $+$ =future, $-$ =past, and

$$\mathcal{M}^+_{\underline{x}_i} = \begin{bmatrix} \pi_{\underline{cx}_i \mapsto \underline{x}_i}(\cdot) \\ \lambda_{\underline{cx}_i \mapsto \underline{x}_i}(\cdot) \end{bmatrix}, \quad \mathcal{M}^-_{\underline{x}_i} = \begin{bmatrix} \pi_{\underline{x}_i \mapsto \underline{px}_i}(\cdot) \\ \lambda_{\underline{x}_i \mapsto \underline{px}_i}(\cdot) \end{bmatrix}. \tag{27.30}$$

Note that

$$\mathcal{M}^-_{\underline{x}_i} = \mathcal{M}^+_{\underline{px}_i} \tag{27.31}$$

for all nodes $\underline{x}_i$. We will refer to Eqs.(27.31) as the **memory overlap conditions**.

We will also use a permuted version of the memory matrix

$$\mathcal{M}'_{\underline{x}_i} = [\mathcal{M}^{OUT}_{\underline{x}_i}, \mathcal{M}^{IN}_{\underline{x}_i}], \tag{27.32}$$

where

$$\mathcal{M}^{OUT}_{\underline{x}_i} = \begin{bmatrix} \pi_{\underline{cx}_i \mapsto \underline{x}_i}(\cdot) \\ \lambda_{\underline{x}_i \mapsto \underline{px}_i}(\cdot) \end{bmatrix}, \quad \mathcal{M}^{IN}_{\underline{x}_i} = \begin{bmatrix} \pi_{\underline{x}_i \mapsto \underline{px}_i}(\cdot) \\ \lambda_{\underline{cx}_i \mapsto \underline{x}_i}(\cdot) \end{bmatrix}. \tag{27.33}$$

Figure 27.6: BP Memory Bnet for the bnet Fig.27.4.

Unfortunately, 2-track bnets cannot be generalized in any obvious way from Markov chains to more complicated DAGs. An alternative to 2-track bnets that still carries message info in its nodes, are memory bnets. An BP **memory bnet** is a bnet which takes each node of an original bnet and adds a local memory to it. More specifically, it keeps tha DAG but replaces each node $\underline{x}_i$ by a memory $\underline{\mathcal{M}}_{\underline{x}_i}$. Fig.27.6 shows the memory bnet for the bnet Fig.27.4. The TPM, printed in blue, for the node $\underline{\mathcal{M}}_{\underline{x}}$ of the memory bnet Fig.27.4, is as follows

$$P(\mathcal{M}_{\underline{x}_i}|\mathcal{M}_{\underline{n}\in nb(\underline{x}_i)}) = AB \ , \tag{27.34}$$

where

$$A = \mathbb{1}(\mathcal{M}^{-}_{\underline{x}_i} = \mathcal{M}^{+}_{\underline{px}_i}) \ , \tag{27.35}$$

and

$$B = \mathbb{1}(\mathcal{M}^{OUT}_{\underline{x}_i} = \mathcal{C}(\mathcal{M}^{IN}_{\underline{x}_i})) \ . \tag{27.36}$$

The function $\mathcal{C}$, which we will call the **BP local computation**, maps $\mathcal{M}^{IN}_{\underline{x}_i}$ into $\mathcal{M}^{OUT}_{\underline{x}_i}$. More explicitly, $\mathcal{C}$ is defined so that

$$B = \underbrace{P(\pi_{\underline{b}\mapsto\underline{x}}|\pi_{\underline{x}\mapsto\underline{a}})}_{B_\pi}\underbrace{P(\lambda_{\underline{x}\mapsto\underline{a}}|\lambda_{\underline{b}\mapsto\underline{x}})}_{B_\pi} \ , \tag{27.37}$$

where B_π and B_λ are given by Eqs.(27.26) and (27.28), respectively.

The BP memory bnet Fig. 27.6 is a deterministic bnet. A deterministic bnet is basically just a coupled system of equations (CSE) for some unknowns x_i. A CSE per se does not include with it a method for solving for the x_i. Such methods are not unique. For example, for the distributed soldier counting problem, the various methods that we described for counting soldiers are just different methods for solving the same CSE. One can describe a method for solving a CSE using a dynamic bnet.[1] To solve the CSE represented by the memory bnet Fig.27.6, we will use the dynamic bnet Fig.27.7. Henceforth, we will refer to Fig.27.7 as an **BP dynamic bnet** for Fig.27.6.

Next, we will explain the meaning of Fig.27.7. Fig.27.7 is a step by step recipe (i.e., algorithm) for solving a SCE, where the unknowns are memory matrices. Each step encoded in Fig.27.7 corresponds to a specific message sending event, where the messages are sent along the edges of the Markov chain Fig.27.4. These message sending events are portrayed in chronological order in Fig.27.8. In that figure, π messages are indicated by dashed red arrows, and λ messages by dotted red arrows. These steps, or message sending events, lead to an updating of the memory matrices

[1] The term dynamic bnet was used in Chapter 13 to mean a time inhomogeneous Markov chain, but here we are stretching its meaning to include Markov chains that aren't time inhomogeneous.

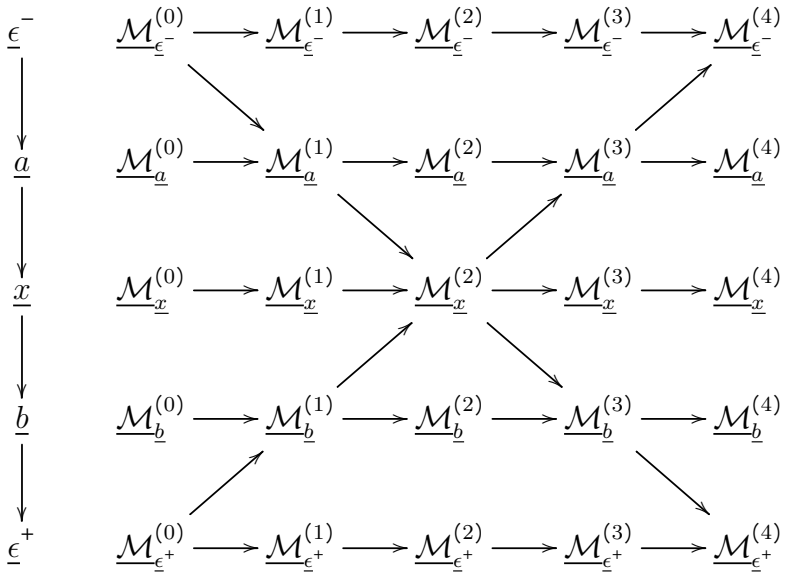


Figure 27.7: BP dynamic bnet for the bnet Fig.27.6.

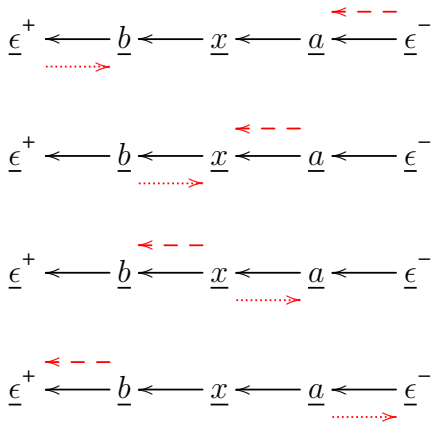


Figure 27.8: Steps encoded in the bnet Fig.27.7. Note the similarity of this figure to Fig.27.1 (d) for soldier counting.

that we are solving for. Each step propagates information between the memory nodes. In the usual Pearl BP algo, the evidence nodes initiate the BP chain of message passing events. These events continue until the memory matrices reach an equilibrium and the SCE is solved.

To use bnet Fig.27.7, we need to specify the **initial conditions** (i.e., the value of $\mathcal{M}_{\underline{x}_i}^{(0)}$ for all i). For that, one can use

$$\pi_{\underline{x}_0}^{(0)} = \delta(x_0, x_0') \ , \tag{27.38}$$

$$\lambda_{\underline{x}_{nx-1}}^{(0)} = \delta(x_{nx-1}, x_{nx-1}') \ . \tag{27.39}$$

All other $\mathcal{M}^{(0)}_{\underline{x}_i}$ entries for all i can be set to 1.

The TPMs, printed in blue, for the nodes of Fig.27.7, can all be summarized by

$$P(\mathcal{M}^{(t)}_{\underline{x}_i}|\mathcal{M}^{(t-1)}_{\underline{n}\in nb(\underline{x}_i)},\mathcal{M}^{(t-1)}_{\underline{x}_i}) = AB \ , \tag{27.40}$$

where

$$A = \begin{cases} \mathbb{1}(\mathcal{M}^{(t)-}_{\underline{x}_i} = \mathcal{M}^{(t-1)+}_{\underline{px}_i}) & \text{if input from } \underline{px}_i \\ \\ \mathbb{1}(\mathcal{M}^{(t)+}_{\underline{x}_i} = \mathcal{M}^{(t-1)-}_{\underline{cx}_i}) & \text{if input from } \underline{cx}_i \end{cases} , \tag{27.41}$$

and

$$B = \mathbb{1}(\mathcal{M}^{(t)OUT}_{\underline{x}_i} = \mathcal{C}(\mathcal{M}^{(t)IN}_{\underline{x}_i})) \ . \tag{27.42}$$

The function $\mathcal{C}$, which we will call the **BP local computation**, maps $\mathcal{M}^{(t)IN}_{\underline{x}_i}$ into $\mathcal{M}^{(t)OUT}_{\underline{x}_i}$. More explicitly, $\mathcal{C}$ is defined so that

$$B = B_\pi B_\lambda \tag{27.43}$$

where

$$B_\pi = \prod_x \mathbb{1}\left(\underbrace{\pi^{(t)}_{\underline{b}\leftarrow\underline{x}}(x)}_{OUT} = \sum_a P(x|a) \underbrace{\pi^{(t)}_{\underline{x}\leftarrow\underline{a}}(a)}_{IN} \right) \tag{27.44}$$

and

$$B_\lambda = \prod_a \mathbb{1}\left(\underbrace{\lambda^{(t)}_{\underline{x}\mapsto\underline{a}}(a)}_{OUT} = \sum_x P(x|a) \underbrace{\lambda^{(t)}_{\underline{b}\mapsto\underline{x}}(x)}_{IN} \right) \ . \tag{27.45}$$

The basic idea behind Eq.(27.42), which we will call the **memory updating equation**, is simple: the memory overlap conditions translate the information from time $t-1$ to t, and then the local computation translates IN to OUT at fixed time t and location $\underline{x}_i$.

BP Algorithm for Polytrees

Consider Fig.27.9, which illustrates a bnet node $\underline{x}$ receiving and sending messages to its neighbors.

Note that in Fig.27.9, the λ or π subscripts $\mapsto$ and $\leftarrow$ indicate the direction of the message.[2] A function is labeled λ (a likelihood function) if the message direction, and the graph arrow, point

[2] I call the symbols $\leftarrow$ and $\mapsto$, a left dart and a right dart. The feathers at the end are intended to evoke a payload of $\pi\&\lambda$ (i.e., past & future) information. On the other hand, I denote the DAG arrows with plain arrows with no feathers because they are causal direction indicators rather than carriers of $\pi\&\lambda$ information.

in opposite directions (i.e, message "swimming upstream", towards the past), whereas a function is labeled π (a probability function) if they point in the same direction (i.e, message "swimming downstream", towards the future). For example, in $\lambda_{\underline{b}\mapsto\underline{x}}(x)$, the message goes from $\underline{b}$ to $\underline{x}$ and the graph arrow points in the opposite direction. In $\pi_{\underline{b}\leftarrow\underline{x}}(x)$, the message goes from $\underline{x}$ to $\underline{b}$ and the graph arrow points in the same direction.

Note that argument arg of the $\pi(arg)$ and $\lambda(arg)$ functions is always the same as the letter in the subscript that is closest to the argument.

Note that in Fig.27.9, we indicate messages that travel "downstream" (resp., "upstream"), by arrows with dashed (resp., dotted) lines as shafts. Mnemonic: think of the shaft as a velocity vector field for the message. You travel faster when you swim downstream as opposed to upstream.

$pa(\underline{x})$ = parents of node $\underline{x}$
$ch(\underline{x})$ = children of node $\underline{x}$
$nb(\underline{x}) = pa(\underline{x}) \cup ch(\underline{x})$ = neighbors of node $\underline{x}$

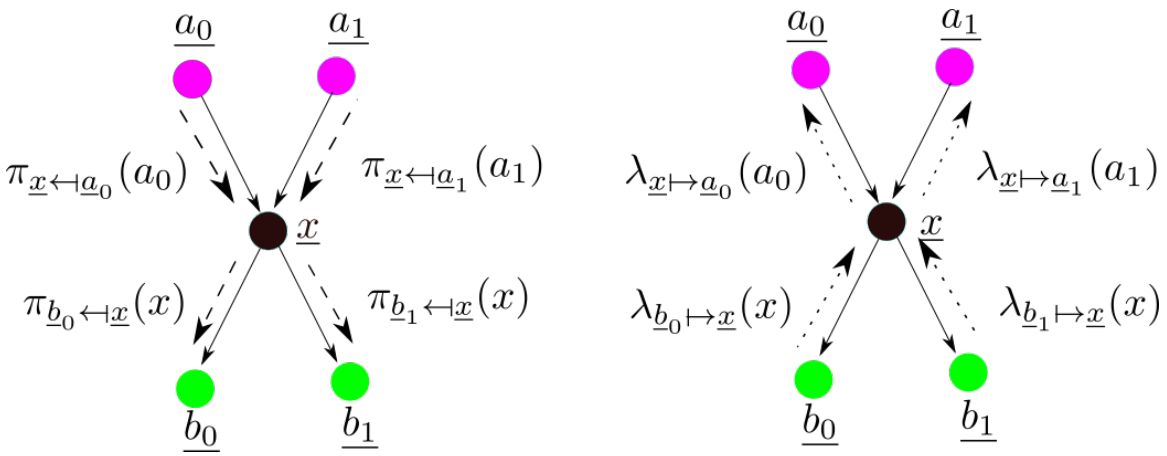


Figure 27.9: Node $\underline{x}$ receiving and sending messages to its neighbors. (neighbors= parents and children).

We define a **memory matrix** $\mathcal{M}_{\underline{x}}$ for node $\underline{x}$ as

$$\mathcal{M}_{\underline{x}} = [\mathcal{M}^+_{\underline{x}}, \mathcal{M}^-_{\underline{x}}] \,, \tag{27.46}$$

where + =future, − =past, and

$$\mathcal{M}^+_{\underline{x}} = \left[\begin{array}{cc} \pi_{\underline{b}\leftarrow\underline{x}}(\cdot) & \lambda_{\underline{b}\mapsto\underline{x}}(\cdot) \end{array} \right]_{\underline{b}\in ch(\underline{x})} = [\mathcal{M}^+_{\underline{b},\underline{x}}]_{\underline{b}\in ch(\underline{x})} \,, \tag{27.47}$$

$$\mathcal{M}^-_{\underline{x}} = \left[\begin{array}{c} \pi_{\underline{x}\leftarrow\underline{a}}(\cdot) \\ \lambda_{\underline{x}\mapsto\underline{a}}(\cdot) \end{array} \right]_{\underline{a}\in pa(\underline{x})} = [\mathcal{M}^-_{\underline{x},\underline{a}}]_{\underline{a}\in pa(\underline{x})} \,. \tag{27.48}$$

Note that

$$\mathcal{M}^-_{\underline{x},\underline{a}} = \mathcal{M}^+_{\underline{a},\underline{x}} \tag{27.49}$$

for every arrow $\underline{x} \leftarrow \underline{a}$. We will refer to Eqs.(27.49) as the **memory overlap conditions**.

We will also use a permuted version of the memory matrix

$$\mathcal{M}'_{\underline{x}} = [\mathcal{M}^{OUT}_{\underline{x}}, \mathcal{M}^{IN}_{\underline{x}}] \,, \tag{27.50}$$

where

$$\mathcal{M}^{OUT}_{\underline{x}} = \begin{pmatrix} [\pi_{\underline{b}\mapsto\underline{x}}(\cdot)]_{\underline{b}\in ch(\underline{x})} \\ [\lambda_{\underline{x}\mapsto\underline{a}}(\cdot)]_{\underline{a}\in pa(\underline{x})} \,, \end{pmatrix} = [\mathcal{M}^{OUT}_{\underline{x},\underline{n}}]_{\underline{n}\in nb(\underline{x})} \,, \tag{27.51}$$

$$\mathcal{M}^{IN}_{\underline{x}} = \begin{pmatrix} [\pi_{\underline{x}\mapsto\underline{a}}(\cdot)]_{\underline{a}\in pa(\underline{x})} \\ [\lambda_{\underline{b}\mapsto\underline{x}}(\cdot)]_{\underline{b}\in ch(\underline{x})} \end{pmatrix} = [\mathcal{M}^{IN}_{\underline{x},\underline{n}}]_{\underline{n}\in nb(\underline{x})} \,. \tag{27.52}$$

For times $t = 0, 1, \ldots, T-1$, we calculate $\mathcal{M}^{(t)}_{\underline{x}}$ in two steps: first we calculate $\mathcal{M}^{(t)IN}_{\underline{x}}$ from earlier memories at time $t-1$, then we calculate $\mathcal{M}^{(t)OUT}_{\underline{x}}$:

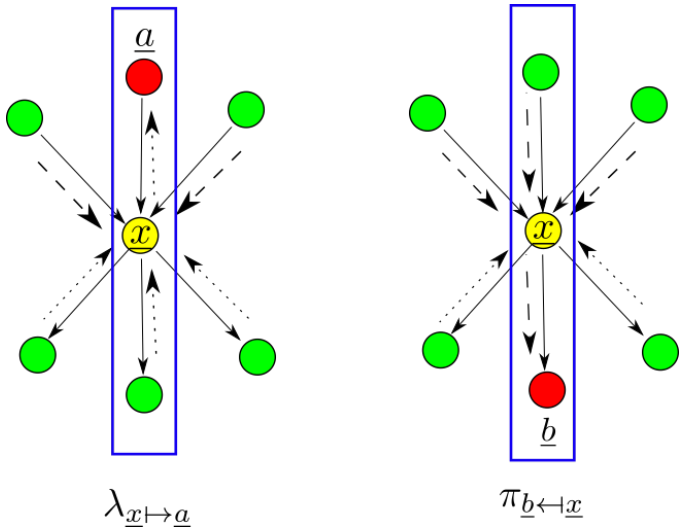


Figure 27.10: Subgraph of a bnet showing two cases (RULE 1 and RULE 2) of message info flow. The yellow node is a gossip monger. It receives messages from all the green nodes, and then it relays a joint message to the red node. Union of green nodes and the red node = full neighborhood of yellow node. There are two possible cases: the red node is either a parent or a child of the yellow one. As usual, we use arrows with dashed (resp., dotted) shafts for downstream (resp., upstream) messages. Blue boxes indicate Markov chain case.

An **evidence node** is a node whose TPM is a delta function set to a particular state of the node. We will assume, without loss of generality, that all evidence nodes are leaf nodes. If that is not the case, any evidence node $\underline{e}$ that is not a leaf node, can be given a new companion leaf node $\underline{l}$ connected to $\underline{e}$ by an arrow $\underline{l} \leftarrow \underline{e}$, and such that $\underline{l}$ has a delta function TPM.

1. **Calculating $\mathcal{M}^{(t)IN}_{\underline{x}}$ from signals received from $\underline{n} \in nb(\underline{x})$, sent at earlier time $t-1$:**

Set

$$\mathcal{M}^{(t)-}_{\underline{x},\underline{a}}|_\pi = \mathcal{M}^{(t-1)+}_{\underline{a},\underline{x}}|_\pi \ , \tag{27.53}$$

for all $\underline{a} \in pa(\underline{x})$, and

$$\mathcal{M}^{(t)+}_{\underline{b},\underline{x}}|_\lambda = \mathcal{M}^{(t-1)-}_{\underline{x},\underline{b}}|_\lambda \ , \tag{27.54}$$

for all $\underline{b} \in ch(\underline{x})$. By $X|_\lambda$ (resp., $X|_\pi$) we mean the λ (resp., π) component of X.

2. **Calculating $\mathcal{M}^{(t)OUT}_{\underline{x}}$ from already calculated $\mathcal{M}^{(t)IN}_{\underline{x}}$:**

Let $\underline{a}^{na} = (\underline{a}_i)_{i=0,1,...,na-1}$ denote the parents of $\underline{x}$ and $\underline{b}^{nb} = (\underline{b}_i)_{i=0,1,...,nb-1}$ its children. Define

$$\pi_{\underline{x}}(x) = \sum_{a^{na}} P(x|a^{na}) \prod_i \pi_{\underline{x} \leftarrow \underline{a}_i}(a_i) \tag{27.55}$$

$$= E_{\underline{a}^{na}}[P(x|a^{na})] \tag{27.56}$$

(boundary case: if $\underline{x}$ is a root node, use $\pi_{\underline{x}}(x) = P(x)$.) and

$$\lambda_{\underline{x}}(x) = \prod_i \lambda_{\underline{b}_i \mapsto \underline{x}}(x) \ . \tag{27.57}$$

(boundary case: if $\underline{x}$ is a leaf node, use $\lambda_{\underline{x}}(x) = 1$.)

- **RULE 1: (red parent)**
From the $\lambda_{\underline{x} \mapsto \underline{a}}$ panel of Fig.27.10, we get[3]

$$\underbrace{\lambda_{\underline{x} \mapsto \underline{a}_i}(a_i)}_{OUT} = \mathcal{N}(!a_i) \sum_x \left[\underbrace{\lambda_{\underline{x}}(x)}_{IN} \sum_{(a_k)_{k \neq i}} \left(P(x|a^{na}) \prod_{k \neq i} \underbrace{\pi_{\underline{x} \leftarrow \underline{a}_k}(a_k)}_{IN} \right) \right] \tag{27.58}$$

$$= \mathcal{N}(!a_i) \sum_x \left[\lambda_{\underline{x}}(x) E_{(\underline{a}_k)_{k \neq i}}[P(x|a^{na})] \right] \tag{27.59}$$

$$= \mathcal{N}(!a_i) E_{(\underline{a}_k)_{k \neq i}} E_{\underline{x}|a^{na}} \lambda_{\underline{x}}(x) \tag{27.60}$$

(boundary case: if $\underline{x}$ is a root node, use $\lambda_{\underline{x} \mapsto \underline{a}_i}(a_i) = \mathcal{N}(!a_i)$.)

- **RULE 2: (red child)**
From the $\pi_{\underline{b} \leftarrow \underline{x}}$ panel of Fig.27.10, we get

$$\underbrace{\pi_{\underline{b}_i \leftarrow \underline{x}}(x)}_{OUT} = \mathcal{N}(!x) \underbrace{\pi_{\underline{x}}(x)}_{IN} \prod_{k \neq i} \underbrace{\lambda_{\underline{b}_k \mapsto \underline{x}}(x)}_{IN} \tag{27.61}$$

(boundary case: if $\underline{x}$ is a leaf node, use $\pi_{\underline{b}_i \leftarrow \underline{x}}(x) = P(x)$.)

[3] As usual in this book, $\mathcal{N}(!x)$ means a constant that is independent of x.

In the above equations, if the range set of a product is empty, then define the product as 1; i.e., $\prod_{k\in\emptyset} F(k) = 1$.

Claim: Define

$$BEL^{(t)}(x) = \mathcal{N}(!x)\lambda_{\underline{x}}^{(t)}(x)\pi_{\underline{x}}^{(t)}(x) \ . \tag{27.62}$$

Then

$$\lim_{t\to\infty} BEL^{(t)}(x) = P(x|\epsilon) \ . \tag{27.63}$$

This says that the belief in $\underline{x} = x$ converges to $P(x|\epsilon)$ and it equals the product of messages received from all parents and children of $\underline{x} = x$.

How BP algo for polytrees reduces to BP algo for Markov chains

It is instructive to see how the BP algo for polytrees reduces to BP algo for Markov chains. For a Markov chain, node $\underline{x}$ has a single parent (i.e., ancestor) $\underline{a}$ and a single child $\underline{b}$. Therefore, Eqs.(27.55) and (27.57) reduce to

$$\pi_{\underline{x}}(x) = \sum_a P(x|a)\pi_{\underline{x}\mapsto\underline{a}}(a) \tag{27.64}$$

and

$$\lambda_{\underline{x}}(x) = \lambda_{\underline{b}\mapsto\underline{x}}(x) \ . \tag{27.65}$$

RULE 1 given by Eq.(27.58) reduces to

$$\begin{aligned} \lambda_{\underline{x}\mapsto\underline{a}}(a) &= \mathcal{N}(!a)\sum_x \lambda_{\underline{x}}(x)P(x|a) &(27.66)\\ &= \mathcal{N}(!a)\sum_x \lambda_{\underline{b}\mapsto\underline{x}}(x)P(x|a) &(27.67)\end{aligned}$$

RULE 2 given by Eq.(27.61) reduces to

$$\begin{aligned} \pi_{\underline{b}\mapsto\underline{x}} &= \mathcal{N}(!x)\pi_{\underline{x}}(x) &(27.68)\\ &= \sum_a P(x|a)\pi_{\underline{x}\mapsto\underline{a}}(a) \ . &(27.69)\end{aligned}$$

Derivation of BP Algorithm for Polytrees

This derivation is taken from the 1988 book Ref.[18] by Judea Pearl, where it is presented very lucidly. We only made some minor changes in notation.

Notation

The BP algorithm yields an expansion for $P(x|\epsilon)$.

$\underline{x}$= the focus node, arbitrary node of bnet that we are focusing on to calculate its $P(x|\epsilon)$.

$(\underline{a}_i)_{i=0,1,\ldots,na-1}$. = parent nodes (mnemonic: a=ancestor) of $\underline{x}$

$(\underline{b}_i)_{i=0,1,\ldots,nb-1}$. = children nodes of $\underline{x}$.

$\underline{\epsilon}$= set of nodes for which there is evidence; that is, $\underline{\epsilon} = \epsilon$, so the state of these nodes is fixed. We always assume $\underline{x} \notin \underline{\epsilon}$ because if $\underline{x} \in \underline{\epsilon}$, and $\underline{\epsilon}$ assigns b to $\underline{x}$, then one can add an arrow coming out of $\underline{x}$ to a new leaf node $\underline{b}$ with TPM $P(b|x) = \delta(b, x)$.

$\underline{\epsilon}^+_{\underline{x}}$= set of evidence nodes that are descendants (in its future) of $\underline{x}$.

$\underline{\epsilon}^-_{\underline{x}}$= set of evidence nodes that are ancestors (in its past) of $\underline{x}$.[4]

$\underline{\epsilon} = \underline{\epsilon}^+_{\underline{x}} \cup \epsilon^-_{\underline{x}}$

$$\pi_{\underline{x}}(x) = P(x|\epsilon^-_{\underline{x}}) \tag{27.70}$$

$$\pi_{\underline{x}\leftarrow\underline{a}_i}(a_i) = \pi(\underline{x} \leftarrow \underline{a}_i = a_i) = \pi(\underline{x} \leftarrow a_i) = P(a_i|\epsilon^-_{\underline{x}\underline{a}_i}) \tag{27.71}$$

$$\pi_{\underline{b}_i\leftarrow\underline{x}}(x) = \pi(\underline{b}_i \leftarrow \underline{x} = x) = \pi(\underline{b}_i \leftarrow x) = P(x|\epsilon^-_{\underline{x}\underline{b}_i}) \tag{27.72}$$

$$\lambda_{\underline{x}}(x) = P(\epsilon^+_{\underline{x}}|x) \tag{27.73}$$

$$\lambda_{\underline{b}_i\mapsto\underline{x}}(x) = \lambda(\underline{b}_i \mapsto \underline{x} = x) = \lambda(\underline{b}_i \mapsto x) = P(\epsilon^+_{\underline{x}\underline{b}_i}|x) \tag{27.74}$$

$$\lambda_{\underline{x}\mapsto\underline{a}_i}(a_i) = \lambda(\underline{x} \mapsto \underline{a}_i = a_i) = \lambda(\underline{x} \mapsto a_i) = P(\epsilon^+_{\underline{x}\underline{a}_i}|a_i) \tag{27.75}$$

Expansions of $\lambda_{\underline{x}}(x)$ and $\pi_{\underline{x}}(x)$ into products of single node messages.

Note that

$$\epsilon^+_{\underline{x}} = \cup_i \epsilon^+_{\underline{x}\underline{b}_i} \ , \tag{27.76}$$

and

$$\epsilon^-_{\underline{x}} = \cup_i \epsilon^-_{\underline{x}\underline{a}_i} \ . \tag{27.77}$$

Therefore,

$$\underbrace{P(x|\epsilon^-_{\underline{x}})}_{\pi_{\underline{x}}(x)} = P(x|\ \cup_i\ \epsilon^-_{\underline{x}\underline{a}_i}) \tag{27.78}$$

$$= \sum_{a^{na}} P(x|a^{na}) P(a^{na}|\ \cup_i\ \epsilon^-_{\underline{x}\underline{a}_i}) \tag{27.79}$$

$$= \sum_{a^{na}} P(x|a^{na}) \prod_i \underbrace{P(a_i|\epsilon^-_{\underline{x}\underline{a}_i})}_{\pi_{\underline{x}\leftarrow\underline{a}_i}(a_i)} \tag{27.80}$$

[4] Careful: Chapter 4 of Ref.[18] uses − indicate the future and + to indicate the past. This is the opposite of us.

$$\underbrace{P(\epsilon^+_{\underline{x}}|x)}_{\lambda_{\underline{x}}(x)} = \prod_i \underbrace{P(\epsilon^+_{\underline{x}\underline{b}_i}|x)}_{\lambda_{\underline{b}_i \mapsto \underline{x}}(x)} \tag{27.81}$$

Note that past and future evidences $\epsilon^-_{\underline{x}}$ and $\epsilon^+_{\underline{x}}$ that are causally connected to $\underline{x}$ are conditionally independent at fixed $\underline{x}$:

$$P(\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}}|x) = P(\epsilon^+_{\underline{x}}|x)P(\epsilon^-_{\underline{x}}|x) \; . \tag{27.82}$$

This observation is key to the proof of the following claim:

Claim 20

$$\begin{aligned} P(x|\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}}) &= P(\epsilon^+_{\underline{x}}|x)P(x|\epsilon^-_{\underline{x}})\frac{1}{P(\epsilon^+_{\underline{x}}|\epsilon^-_{\underline{x}})} && (27.83)\\ &= \mathcal{N}(!x)P(\epsilon^+_{\underline{x}}|x)P(x|\epsilon^-_{\underline{x}}) && (27.84)\\ &= \mathcal{N}(!x) \quad (\epsilon^+_{\underline{x}} \leftarrow x \leftarrow \epsilon^-_{\underline{x}}) && (27.85)\\ &= \mathcal{N}(!x)\lambda_{\underline{x}}(x)\pi_{\underline{x}}(x) && (27.86) \end{aligned}$$

proof:

$$\begin{aligned} P(x|\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}}) &= P(\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}}|x)\frac{P(x)}{P(\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}})} && (27.87)\\ &= P(\epsilon^+_{\underline{x}}|x)P(\epsilon^-_{\underline{x}}|x)\frac{P(x)}{P(\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}})} && (27.88)\\ &= P(\epsilon^+_{\underline{x}}|x)P(x|\epsilon^-_{\underline{x}})\frac{P(\epsilon^-_{\underline{x}})}{P(\epsilon^+_{\underline{x}}, \epsilon^-_{\underline{x}})} && (27.89)\\ &= P(\epsilon^+_{\underline{x}}|x)P(x|\epsilon^-_{\underline{x}})\frac{1}{P(\epsilon^+_{\underline{x}}|\epsilon^-_{\underline{x}})} && (27.90) \end{aligned}$$

QED

Next we prove BP rules 1 and 2.

- **RULE 1 (red parent)**

  Note that

$$\begin{aligned} \epsilon^+_{\underline{x}} \cup \cup_{k\neq i}\epsilon^-_{\underline{x}\underline{a}_k} &= (\epsilon^+_{\underline{x}} \cup \epsilon^-_{\underline{x}}) - \epsilon^-_{\underline{x}\underline{a}_i} && (27.91)\\ &= \epsilon^+_{\underline{x}\underline{a}_i} && (27.92) \end{aligned}$$

  Let $y = (a_k)_{k\neq i}$ and $\epsilon^-_{\underline{y}} = (\epsilon^-_{\underline{x}\underline{a}_k})_{k\neq i}$.

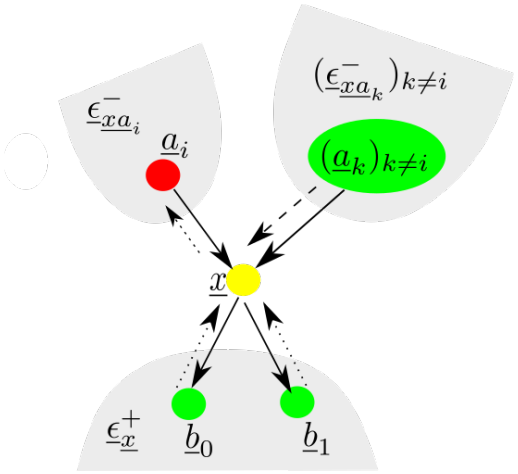


Figure 27.11: This figure is used in the derivation of the BP RULE 1.

$$\underbrace{P(\underline{\epsilon}^+_{\underline{x}\underline{a}_i}|a_i)}_{\lambda_{\underline{x}\mapsto\underline{a}_i}(a_i)} = P(\underline{\epsilon}^+_{\underline{x}}, \underline{\epsilon}^-_{\underline{y}}|a_i) \tag{27.93}$$

$$= \sum_x \sum_y P(\underline{\epsilon}^+_{\underline{x}}, \underline{\epsilon}^-_{\underline{y}}|x,y)P(x,y|a_i) \tag{27.94}$$

$$= \sum_x \sum_y P(\underline{\epsilon}^+_{\underline{x}}|x)P(\underline{\epsilon}^-_{\underline{y}}|y)P(x|y,a_i)P(y|a_i) \tag{27.95}$$

$$= P(\underline{\epsilon}^-_{\underline{y}}) \sum_x \sum_y P(\underline{\epsilon}^+_{\underline{x}}|x) \frac{P(y|\underline{\epsilon}^-_{\underline{y}})}{P(y)} P(x|y,a_i) \underbrace{P(y|a_i)}_{=P(y)} \tag{27.96}$$

$$= \mathcal{N}(!a_i) \sum_x \sum_y P(\underline{\epsilon}^+_{\underline{x}}|x)P(x|\underbrace{y,a_i}_{a^{na}})P(y|\underline{\epsilon}^-_{\underline{y}}) \tag{27.97}$$

$$= \mathcal{N}(!a_i) \sum_x \underbrace{P(\underline{\epsilon}^+_{\underline{x}}|x)}_{\lambda_{\underline{x}}(x)} \sum_{(a_k)_{k\neq i}} P(x|a^{na}) \prod_{k\neq i} \underbrace{P(a_k|\underline{\epsilon}^-_{\underline{x}\underline{a}_k})}_{\pi_{\underline{x}\leftarrow\underline{a}_k}(a_k)} \tag{27.98}$$

- **RULE 2 (red child)**
  Note that

$$(\cup_{k\neq i}\underline{\epsilon}^+_{\underline{x}\underline{b}_k}) \cup \underline{\epsilon}^-_{\underline{x}} = (\underline{\epsilon}^+_{\underline{x}} \cup \underline{\epsilon}^-_{\underline{x}}) - \underline{\epsilon}^+_{\underline{x}\underline{b}_i} \tag{27.99}$$

$$= \underline{\epsilon}^-_{\underline{x}\underline{b}_i} \tag{27.100}$$

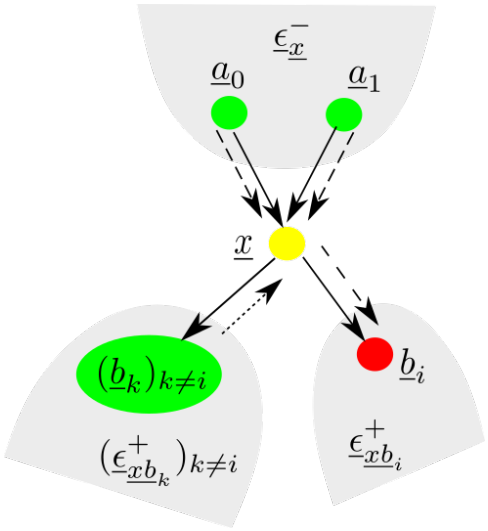


Figure 27.12: This figure is used in the derivation of the BP RULE 2.

$$\underbrace{P(x|\epsilon^-_{\underline{x}\underline{b}_i})}_{\pi_{\underline{b}_i\hookrightarrow\underline{x}}(x)} = P(x|(\epsilon^+_{\underline{x}\underline{b}_k})_{k\neq i}, \epsilon^-_{\underline{x}}) \tag{27.101}$$

$$= \mathcal{N}(!x)P((\epsilon^+_{\underline{x}\underline{b}_k})_{k\neq i}|x)P(x|\epsilon^-_{\underline{x}}) \tag{27.102}$$

$$= \mathcal{N}(!x)\left(\prod_{k\neq i}\underbrace{P(\epsilon^+_{\underline{x}\underline{b}_k}|x)}_{\lambda_{\underline{b}_k\mapsto\underline{x}}(x)}\right)\underbrace{P(x|\epsilon^-_{\underline{x}})}_{\pi_{\underline{x}}(x)} \tag{27.103}$$

Example of BP algo for a Tree

In this section, we describe how to apply the BP algo to the tree bnet Fig.27.13. In Fig.27.13, if we replace each integer i by the random variable $\underline{A}_i$, we get an **original bnet**, and if we replace each i by $\underline{\mathcal{M}}_{\underline{A}_i}$, we get the **BP memory bnet** of the original bnet. In Fig.27.13, the fushia nodes are evidence nodes and the green ones aren't.

We want to solve for the memory matrices of the memory bnet. To do so, we use the **BP dynamic bnet** Fig.27.14. The steps encoded in the dynamic bnet are shown in Fig.27.15. Fig.27.15 has frames in chronological order, showing the direction of travel of the $\pi\&\lambda$ information. This sequence of frames also indicates the order in which we solve for the entries of the memory matrices. The information first emanates from the evidence nodes. It propagates generally upstream, although some nodes can generate downstream flow. Some of the info reaches the root node and is reflected there. The root node is the only one that is capable of reflection (i.e., instant output along an arrow, in response to input along that arrow). Eventually, all info reaches the leaf nodes via downstream propagation and is absorbed there.

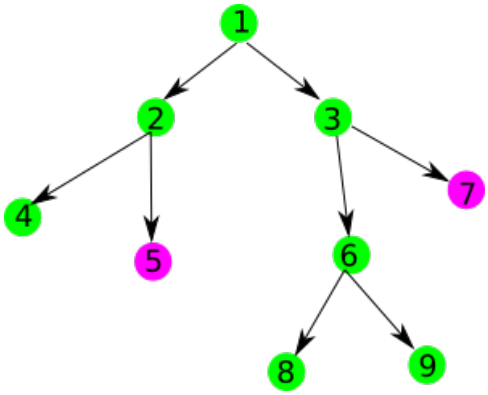


Figure 27.13: Example tree bnet used to illustrate BP.

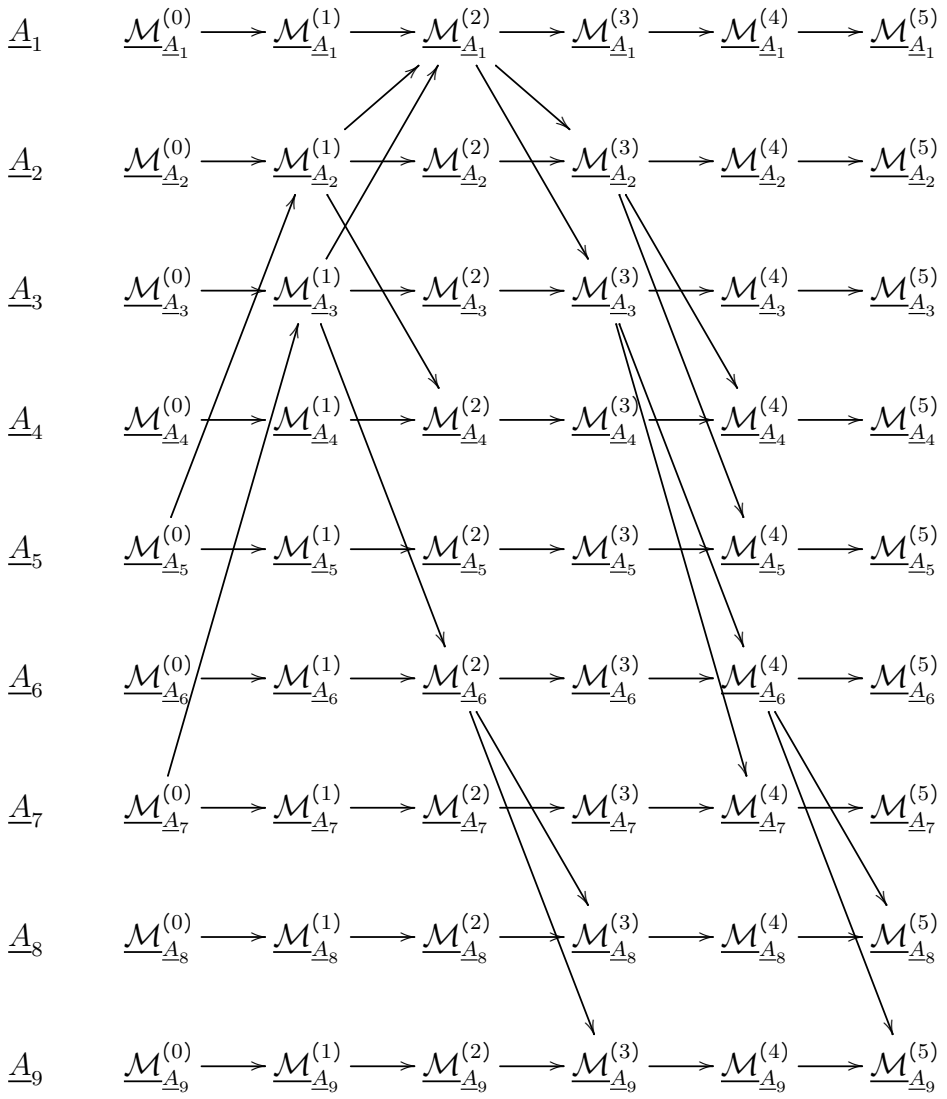


Figure 27.14: BP dynamic bnet for the bnet Fig.27.13.

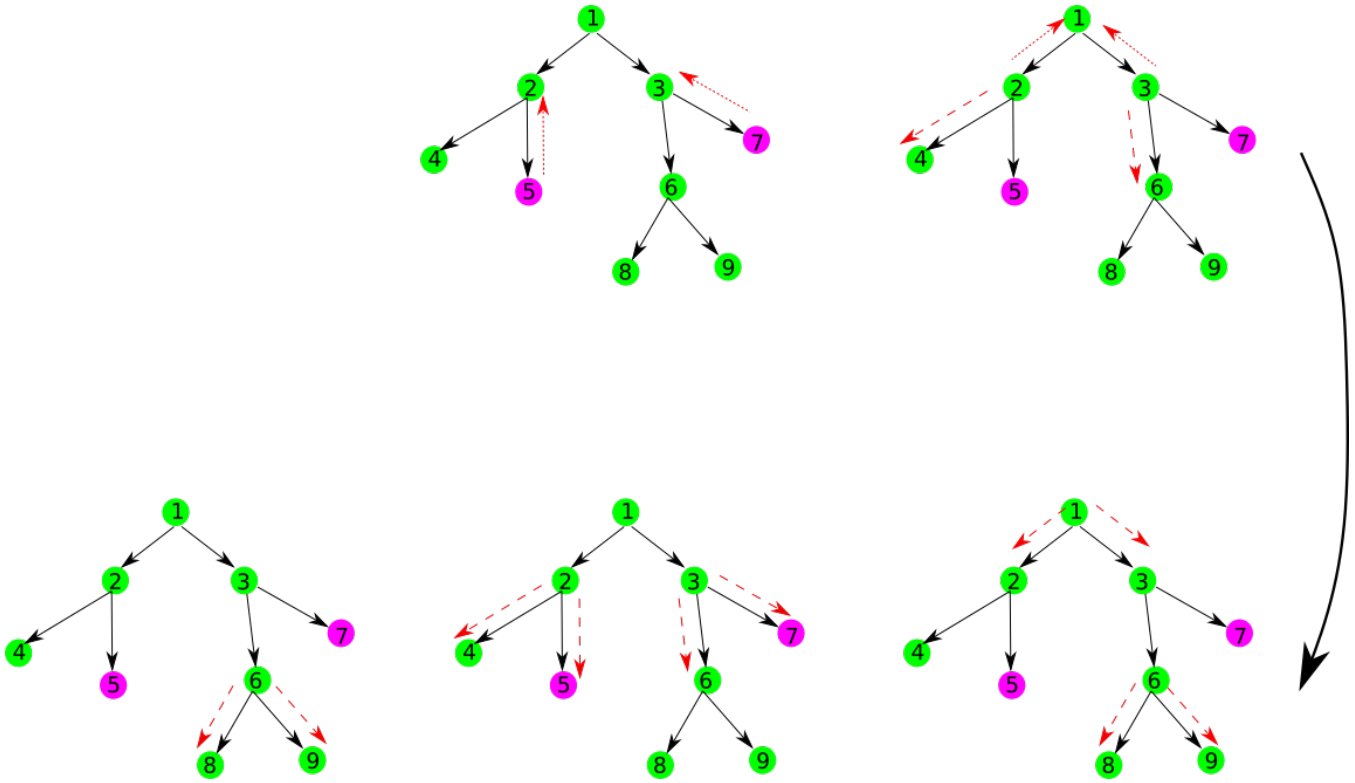


Figure 27.15: Steps encoded in the bnet Fig.27.14.

Bipartite bnets

By a **bipartite bnet** we will mean a bnet in which all nodes are either root nodes (parentless) or leaf nodes (childless). BP simplifies when dealing with bipartite bnets. In this section, we will explain how it simplifies. But before doing so, let us explain how the following two types of diagrams can be replaced by equivalent bipartite bnets:

- Factor Graphs
- Tree bnets

Consider a product $g = \prod_\alpha f_\alpha$ of scalar functions $f_\alpha : S_{\underline{x}_0} \times S_{\underline{x}_1} \times \ldots S_{\underline{x}_{nx-1}} \rightarrow \mathbb{R}$ for $\alpha = 0, 1, \ldots nf-1$. For instance, consider $g : S_{\underline{x}_0} \times S_{\underline{x}_1} \times S_{\underline{x}_2} \rightarrow \mathbb{R}$ defined by:

$$g(x_0, x_1, x_2) = f_0(x_0) f_1(x_0, x_1) f_2(x_0, x_1) f_3(x_1, x_2) \; . \tag{27.104}$$

The **factor graph** for this function g is given by Fig.27.16.

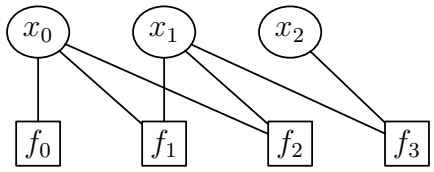


Figure 27.16: Factor graph for function g defined by Eq.(27.104).

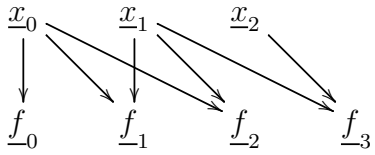


Figure 27.17: Bipartite bnet corresponding to factor graph Fig.27.16.

One can map any factor graph (the "source") to a special bipartite bnet (the "image"), as follows. Replace each x_i by $\underline{x}_i \in S_{\underline{x}_i}$ for $i = 0, 1, \ldots, nx-1$ and each f_α by $\underline{f}_\alpha$ for $\alpha = 0, 1, \ldots, nf-1$. Then replace the connections (edges) of the factor graph by arrows from $\underline{x}_i$ to $\underline{f}_\alpha$. For example, Fig.27.17 is the image bipartite bnet of the source factor graph Fig.27.16.

Let $\underline{x}^{nx} = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{nx-1})$ and $\underline{f}^{nf} = (\underline{f}_0, \underline{f}_1, \ldots, \underline{f}_{nf-1})$. Let[5] $f_\alpha \in \{0, 1\}$ for all α, and $y_\alpha = f_\alpha(x_{nb(\underline{f}_\alpha)})$. Here we are using $nb(\underline{f}_\alpha)$ to denote the neighborhood of node $\underline{f}_\alpha$ in the image

[5] Note that we are using f_α to denote both a function $f_\alpha(\cdot)$ and a boolean value. Which one we mean will be clear from context. f_α could also used to denote, not a function or a boolean value, but the real number $y_\alpha = f_\alpha(x_{nb(\underline{f}_\alpha)})$, but we won't be using it that way in this chapter.

bipartite bnet, and we are using x_S to denote $(x_i)_{i \in S}$. Without loss of generality, we will assume that $y_\alpha \in [0,1]$ for all α. Then we define the node TPMs, printed in blue, for the image bipartite bnet, to be

$$P(f_\alpha | x_{nx(\underline{f}_\alpha)}) = y_\alpha \delta(f_\alpha, 1) + [1 - y_\alpha] \delta(f_\alpha, 0) \tag{27.105}$$

for $\alpha = 0, 1, \ldots, nf - 1$ and

$$P_{\underline{x}_i}(x_i) = \text{arbitrary prior} \tag{27.106}$$

for $i = 0, 1, \ldots, nx - 1$.

Note that

$$P(f^{nf} = 1^{nf} | x^{nx}) = \prod_\alpha f_\alpha(x_{nb(\underline{f}_\alpha)}) \; . \tag{27.107}$$

A **tree bnet** is a bnet for which all nodes have exactly one parent except for the apex root node which has none. A tree bnet is very much like the filing system in a computer.

One can map a tree bnet (the "source") into an equivalent bipartite bnet (the "image") as follows. Replace each arrow

$$\underline{x} \longrightarrow \underline{y} \tag{27.108}$$

of the tree bnet by

$$\underline{x} \longrightarrow \underline{P_{y|x}} \longleftarrow \underline{y} \; . \tag{27.109}$$

For example, the tree bnet Fig.27.18 has the image bipartite bnet given by Fig.27.19. The bnet Fig.27.20 is just a different way of drawing the bnet Fig.27.19.

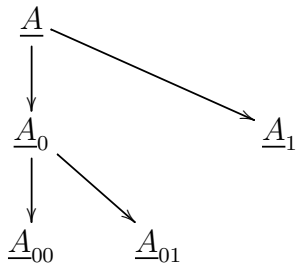


Figure 27.18: Example of a tree bnet.

The node TPMs, printed in blue, for the image bipartite bnet Fig.27.19, are as follows. We express the TPMs of the image bnet in terms of the TPMs of the source bnet Fig.27.18. Let

$$P(P_{\underline{y}|\underline{x}} | x, y) = P_{\underline{y}|\underline{x}}(y|x) \delta(P_{\underline{y}|\underline{x}}, 1) + (1 - P_{\underline{y}|\underline{x}}(y|x)) \delta(P_{\underline{y}|\underline{x}}, 0) \tag{27.110}$$

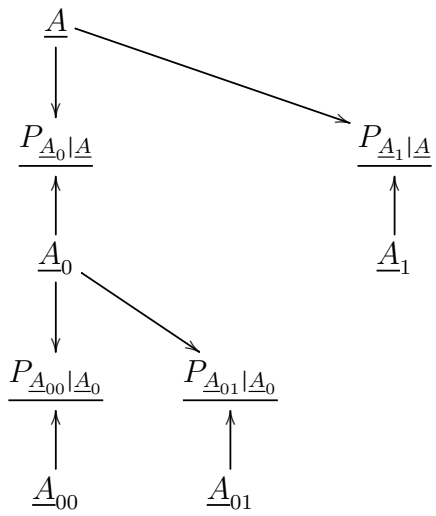


Figure 27.19: Bipartite bnet corresponding to tree bnet Fig.27.18.

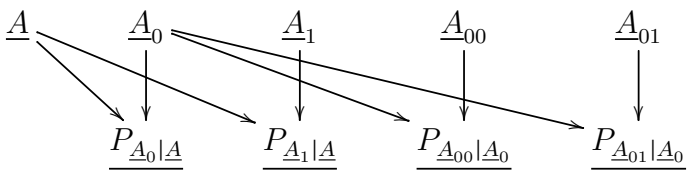


Figure 27.20: Different way of drawing the bnet Fig.27.19.

for all the leaf nodes $\underline{P_{y|x}} \in \{0, 1\}$ of the image bipartite bnet. Also, let

$$P_{\underline{y}}(y) = \text{arbitrary prior} \tag{27.111}$$

for all the root nodes $\underline{y}$ of the image bipartite bnet except when $\underline{y}$ corresponds to the root node $\underline{A}$ of the source tree bnet. In that exceptional case,

$$P_{\underline{y}}(y) = P_{\underline{A}}(y) \; . \tag{27.112}$$

BP for bipartite bnets (BP-BB)

For a bipartite bnet as defined above, with root nodes $\underline{x}_i$ and leaf nodes $\underline{f}_\alpha$, let

$$nb(i) = \{\alpha : \underline{f}_\alpha \in nb(\underline{x}_i)\} \; , \tag{27.113}$$

$$nb(\alpha) = \{i : \underline{x}_i \in nb(\underline{f}_\alpha)\} \ , \tag{27.114}$$

$$m_{\alpha \leftarrow i}(x_i) = \pi_{\underline{f}_\alpha \leftarrow \underline{x}_i}(x_i) \ , \tag{27.115}$$

$$m_{\alpha \mapsto i}(x_i) = \lambda_{\underline{f}_\alpha \mapsto \underline{x}_i}(x_i) \ , \tag{27.116}$$

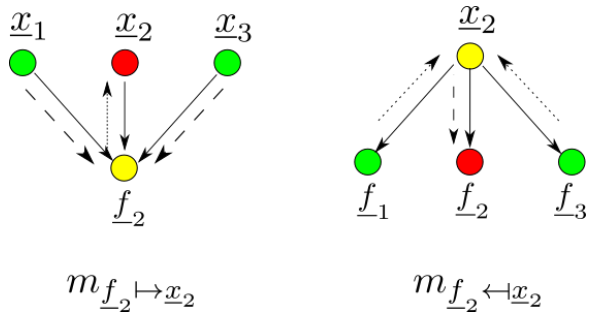


Figure 27.21: Fig.27.10 becomes this figure for the special case of a bipartite bnet. Union of green nodes and the red node = full neighborhood of yellow node. There are two possible cases: the red node is either a parent or a child of the yellow node.

Next we will show how to find $m^{(t)}_{\alpha \leftarrow i}$ and $m^{(t)}_{\alpha \mapsto i}$ from $m^{(t-1)}_{\alpha \leftarrow i}$ and $m^{(t-1)}_{\alpha \mapsto i}$.

1. **Traversing an x (i.e., root) node.**

   See the $m_{\underline{f}_2 \leftarrow \underline{x}_2}$ panel of Fig.27.21.

   For $i = 0, 1, \ldots, nx - 1$, if $\alpha \in nb(i)$, then,

$$m^{(t)}_{\alpha \leftarrow i}(x_i) = \prod_{\beta \in nb(i) - \alpha} m^{(t-1)}_{\beta \mapsto i}(x_i) \ , \tag{27.117}$$

   whereas if $\alpha \notin nb(i)$

$$m^{(t)}_{\alpha \leftarrow i}(x_i) = m^{(t-1)}_{\alpha \leftarrow i}(x_i) \ . \tag{27.118}$$

2. **Traversing an f (i.e., leaf) node.**

   See the $m_{\underline{f}_2 \mapsto \underline{x}_2}$ panel of Fig.27.21.

   For $\alpha = 0, 1, \ldots, nf - 1$, if $i \in nb(\alpha)$, then

$$m^{(t)}_{\alpha \mapsto i}(x_i) = \sum_{(x_k)_{k \in nb(\alpha) - i}} f_\alpha(x_{nb(\alpha)}) \prod_{k \in nb(\alpha) - i} m^{(t-1)}_{\alpha \leftarrow k}(x_k) \tag{27.119}$$

$$= E^{(t-1)}_{(x_k)_{k \in nb(\alpha) - i}} [f_\alpha(x_{nb(\alpha)})] \ , \tag{27.120}$$

whereas if $i \notin nb(\alpha)$

$$m^{(t)}_{\alpha \mapsto i}(x_i) = m^{(t-1)}_{\alpha \mapsto i}(x_i) \ . \tag{27.121}$$

In the above equations, if the range set of a product is empty, then define the product as 1; i.e., $\prod_{k \in \emptyset} F(k) = 1$.

Claim:

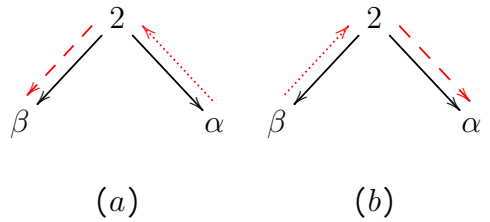
$$P(x_i|\epsilon) = \lim_{t \to \infty} \mathcal{N}(!x_i) \prod_{\alpha \in nb(i)} m^{(t)}_{\alpha \mapsto i}(x_i) \tag{27.122}$$

and

$$P(x_{nb(\alpha)}|\epsilon) = \lim_{t \to \infty} \mathcal{N}(!x_{nb(\alpha)}) f_\alpha(x_{nb(\alpha)}) \prod_{k \in nb(\alpha)} m^{(t)}_{\alpha \leftarrow k}(x_k) \ . \tag{27.123}$$

BP-BB and general BP agree on Markov chains

It is instructive to compare the belief values (i.e., $P(x_i|\epsilon)$) obtained by applying the general (i.e., polytree) BP and BP-BB algorithms to a Markov chain. Next we show that both algorithms yield the same belief values.



(a) (b)

Figure 27.22: Traversing a root node of a Markov chain (a)Propagation towards left (i.e., towards future). (b)Propagation towards right (i.e., towards past).

Consider the BP-BB rule for traversing a root node. When traveling towards the left as in Fig.27.22 (a), it implies that

$$m_{\alpha \mapsto 2}(x_2) = m_{\beta \leftarrow 2}(x_2) \ , \tag{27.124}$$

and when traveling towards the right as in Fig.27.22 (b), it implies that

$$m_{\beta \mapsto 2}(x_2) = m_{\alpha \leftarrow 2}(x_2) \ . \tag{27.125}$$

Now consider the BP-BB rule for traversing a leaf node. When traveling to the left as in Fig.27.23 (a), it implies that

$$\underbrace{m_{\alpha \mapsto 2}(x_2)}_{\lambda} = \sum_{x_1} P(x_2|x_1) \underbrace{m_{\alpha \leftarrow 1}(x_1)}_{\pi} \ . \tag{27.126}$$

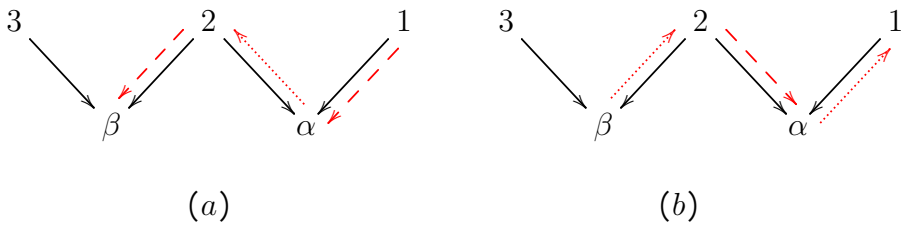


Figure 27.23: Traversing a leaf node of a Markov chain (a)Propagation towards left (i.e., towards future). (b)Propagation towards right (i.e., towards past).

One can rewrite the left and right hand sides (LHS, RHS) of Eq.(27.126) as follows

$$RHS = \sum_{x_1} P(x_2|x_1)\pi_{\alpha\leftarrow 1}(x_1) \ , \tag{27.127}$$

and

$$LHS = m_{\alpha\rightarrow 2}(x_2) = m_{\beta\leftarrow 2}(x_2) = \pi_{\beta\leftarrow 2}(x_2) \ , \tag{27.128}$$

Therefore

$$\pi_{\beta\leftarrow 2}(x_2) \sum_{x_1} P(x_2|x_1)\pi_{\alpha\leftarrow 1}(x_1) \ . \tag{27.129}$$

Once again, consider the BP-BB rule for traversing a leaf node. When traveling to the right as in Fig.27.23 (b), it implies that

$$\underbrace{m_{\alpha\rightarrow 1}(x_1)}_{\lambda} = \sum_{x_2} P(x_2|x_1) \underbrace{m_{\alpha\leftarrow 2}(x_2)}_{\pi} \ . \tag{27.130}$$

One can rewrite the left and right hand sides (LHS, RHS) of Eq.(27.130) as follows

$$RHS = \ = \sum_{x_2} P(x_2|x_1)\pi_{\alpha\leftarrow 2}(x_2) \tag{27.131}$$

$$= \sum_{x_2} P(x_2|x_1)\lambda_{\beta\rightarrow 2}(x_2) \ , \tag{27.132}$$

and

$$LHS = \lambda_{\alpha\rightarrow 1}(x_1) \ . \tag{27.133}$$

Therefore,

$$\lambda_{\alpha\rightarrow 1}(x_1) = \sum_{x_2} P(x_2|x_1)\lambda_{\beta\rightarrow 2}(x_2) \ . \tag{27.134}$$

Finally, note that

$$\begin{aligned} P(x_2|\epsilon) &= \mathcal{N}(!x_2)m_{\beta\mapsto 2}(x_2)m_{\alpha\mapsto 2}(x_2) && (27.135)\\ &= \mathcal{N}(!x_2)m_{\alpha\leftarrow 2}(x_2)m_{\alpha\mapsto 2}(x_2) && (27.136)\\ &= \mathcal{N}(!x_2)\pi_{\alpha\leftarrow 2}(x_2)\lambda_{\alpha\mapsto 2}(x_2) && (27.137)\\ &= \mathcal{N}(!x_2)P(x_2|\epsilon^-)P(x_2|\epsilon^+) && (27.138) \end{aligned}$$

and

$$\begin{aligned} P(x_2,x_1) &= \mathcal{N}(!x_2,!x_1)P(x_2|x_1)m_{\alpha\leftarrow 1}(x_1)m_{\alpha\leftarrow 2}(x_2) && (27.139)\\ &= \mathcal{N}(!x_2,!x_1)P(x_2|x_1)\pi_{\alpha\leftarrow 1}(x_1)\pi_{\alpha\leftarrow 2}(x_2)\ . && (27.140) \end{aligned}$$

BP-BB and general BP agree on tree bnets.

It is instructive to compare the belief values (i.e., $P(x_i|\epsilon)$) obtained by applying the general (i.e., polytree) BP and BP-BB algorithms to a tree bnet. Next we show that both algorithms yield the same belief values.

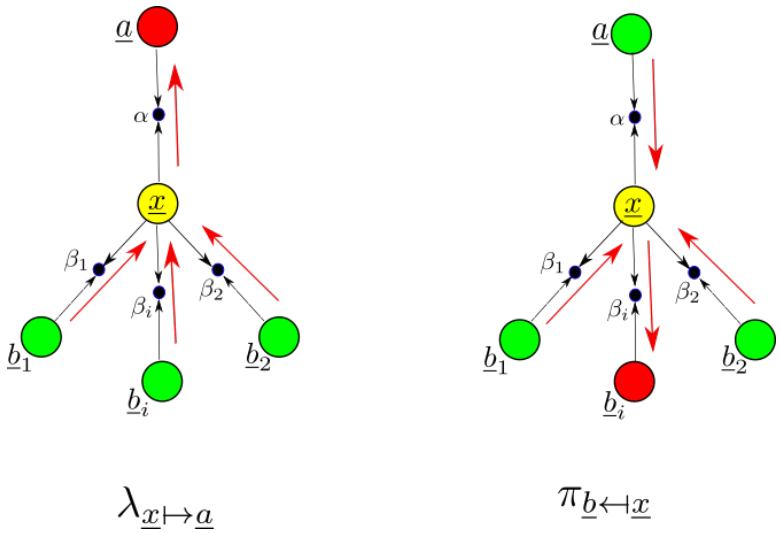


Figure 27.24: Subgraph of a tree bnet. This is the same as Fig.27.10, except that here the yellow node has a single parent because this is a subgraph of a tree bnet, not of an arbitrary bnet like Fig.27.10. The subgraph has been converted to a subgraph of a bipartite bnet by inserting a collider leaf node, labeled by a Greek letter, at the center of each edge of the tree bnet. Red arrows indicate the direction of message info flow.

Applying to the left panel of Fig.27.24 the BP-BB rule for traversing a root node, we get

$$m_{\alpha\leftarrow\underline{x}}(x) = \prod_i m_{\beta_i\mapsto\underline{x}}(x)\ . \qquad (27.141)$$

Applying to the left panel of Fig.27.24 the BP-BB rule for traversing a leaf node, we get

$$m_{\alpha\mapsto\underline{a}}(a) = \mathcal{N}(!a)\sum_x m_{\alpha\leftarrow\underline{x}}(x)P(x|a) \ . \tag{27.142}$$

Combining Eqs.(27.141) and (27.142), we get

$$m_{\alpha\mapsto\underline{a}}(a) = \mathcal{N}(!a)\sum_x P(x|a)\prod_i m_{\beta_i\mapsto\underline{x}}(x) \ , \tag{27.143}$$

which can be rewritten as

$$\lambda_{\underline{x}\mapsto\underline{a}}(a) = \mathcal{N}(!a)\sum_x P(x|a)\underbrace{\prod_i \lambda_{\underline{b}_i\mapsto\underline{x}}(x)}_{\lambda_{\underline{x}}(x)} \ . \tag{27.144}$$

Eq.27.144 is just RULE 1 for general BP.

Applying to the right panel of Fig.27.24 the BP-BB rule for traversing a root node, we get

$$m_{\beta_i\leftarrow\underline{x}}(x) = \mathcal{N}(!x)m_{\alpha\mapsto\underline{x}}(x)\prod_{k\neq i} m_{\beta_k\mapsto\underline{x}}(x) \tag{27.145}$$

Applying to the right panel of Fig.27.24 the BP-BB rule for traversing a leaf node, we get

$$\begin{aligned} m_{\alpha\mapsto\underline{x}}(x) &= \sum_a P(x|a)m_{\alpha\leftarrow a}(a) && (27.146)\\ &= \sum_a P(x|a)\pi_{\underline{x}\leftarrow a}(a) && (27.147)\\ &= \pi_{\underline{x}}(x) \ . && (27.148) \end{aligned}$$

Combining Eqs.(27.145) and (27.148), we get

$$\pi_{\underline{b}_i\leftarrow\underline{x}}(x) = \mathcal{N}(!x)\pi_{\underline{x}}(x)\prod_{k\neq i}\lambda_{\underline{b}_k\mapsto\underline{x}}(x) \ . \tag{27.149}$$

Eq.(27.149) is just RULE 2 of general BP.

BP-BB and sum-product decomposition

BP-BB yields what is often referred to as a **sum-product decomposition**. I don't like that name because it is unnecessarily confusing, and it fails to convey the recursive nature[6] of the decomposition. I prefer to call it a **recursive sum of products (RSOP) decomposition**, and will call it so henceforth in this chapter.

[6] By "recursive nature", we mean bootstraped definitions that lead to nested sums. The recursive nature of BP is evident from RULES 1 and 2 that define λ's and π's in terms of other λ's and π's.

Expressing the marginals of a bnet as RSOPs, which is what BP does, reduces the complexity of the calculation. (i.e., the total number of additions and multiplications that need to be performed) That makes using the BP algo very advantageous. For instance, consider a Markov chain $\underline{x}_{n-1} \leftarrow \cdots \leftarrow \underline{x}_1 \leftarrow \underline{x}_0$. Note that if one calculates $P(x_{n-1})$ as follows

$$P(x_{n-1}) = \left[\sum_{x_{n-2}} P(x_{n-1}|x_{n-2}) \ldots \left[\sum_{x_1} P(x_2|x_1) \left[\sum_{x_0} P(x_1|x_0)P(x_0) \right] \right] \ldots \right] , \tag{27.150}$$

we need to perform $2(n-1)$ additions and $3(n-1)$ multiplications. On the other hand, if we calculate $P(x_{n-1})$ as follows

$$P(x_{n-1}) = \sum_{x_{n-2}} \ldots \sum_{x_1} \sum_{x_0} P(x_{n-1}|x_{n-2}) \ldots P(x_2|x_1)P(x_1|x_0)P(x_0) , \tag{27.151}$$

we must perform $3^n - 1$ additions and $3^n(n-1)$ multiplications.

Chapter 28

Missing Data, Imputation

This chapter assumes that the reader has read some parts of Chapter 14 on the Expectation Maximization (EM) algo and Chapter 25 on Markov Chain Monte Carlo (MCMC).

	h_0	x_0	x_1	x_2
1	NA	0	1	1
2	NA	0	0	0
3	NA	1	1	0
4	NA	NA	1	NA
5	NA	0	NA	1
6	NA	0	0	1

	h_0	x_0	x_1	x_2	m
1	NA	0	1	1	(0,0,0)
2	NA	0	0	0	(0,0,0)
3	NA	1	1	0	(0,0,0)
4	NA	0 0 1 1	1	0 1 0 1	(1,0,1)
5	NA	0	0 1	1	(0,1,0)
6	NA	0	0	1	(0,0,0)

Table 28.1: **Left Table:** Dataset with $nsam = 6$ and some missing entries, for 4 binary variables h_0, x_0, x_1, x_2. NA=not available. The h_0 column is completely missing because h_0 is an unobserved latent variable. **Right Table:** All possibilities for $x_i = NA$ cells of left table have been enumerated. A new column labeled m has been added. $m_i = \mathbb{1}(x_i \text{ is missing})$ for $i = 0, 1, 2$.

Suppose that you have compiled a **dataset** $\vec{x} = (x[\sigma])_{\sigma=0,1,\ldots,nsam-1}$ where $x = (x_0, x_1, \ldots, x_{nx-1})$ from a study or survey. It consists of $nsam$ number of samples (sample= row), and nx columns (each column is a different feature, or observation). Suppose that some of the cells in this matrix are empty. Throwing away all the incomplete rows is okay if the number of incomplete rows is much smaller than $nsam$. If not, throwing them away would throw away a substantial amount of information contained in all the filled cells in those incomplete rows, plus it might bias your dataset. This chapter deals with how to fill those empty cells with plausible fake data. A fancy name for this process is **imputation**. There is no unique way of fabricating fake data, but some fakes are better than others by some metrics. This chapter will consider two popular ways (EM and MCMC) of filling those empty cells with their "most likely" values based on the cells of the dataset that aren't missing, and also based on some bnet model that is expected to describe well the dataset.

Notation: $\underline{\vec{a}} = (\underline{a}[\sigma])_{\sigma=0,1,...,nsam-1}$, where $nsam$ is the number of samples. Will sometimes denote $a[\sigma]$ by $a^{[\sigma]}$.

For concreteness, we will apply the concepts of this chapter to the dataset with missing data given by Table 28.1.

Imputation via EM

We begin by augmenting Fig.14.1 (the first figure in Chapter 14) by adding to it a new node $\underline{\vec{m}}$ called the **missingness variable**. Recall that node $\underline{\theta}$ represents the **unknown parameters**, node $\underline{\vec{x}}$ represents the **observed variables**, and node $\underline{\vec{h}}$ represents the **latent variables**. Both $\underline{\theta}$ and $\underline{\vec{h}}$ are hidden (i.e., unobserved). Fig.28.1 shows 3 popular ways of connecting node $\underline{\vec{m}}$ to the other nodes in the graph Fig.14.1.

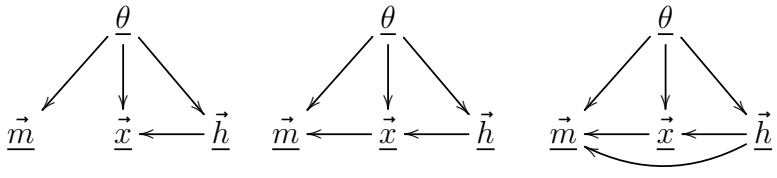


Figure 28.1: The left bnet is seldom assumed. The middle bnet is referred to as the MAR (missing at random) assumption. The right bnet is referred to as the not-MAR (NMAR) assumption.

From Fig.28.1, we have

$$P(\vec{m}|\vec{x},\vec{h},\theta) = \begin{cases} P(\vec{m}|\theta) & \text{Seldom assumed. Called missing-CAR (MCAR)} \\ P(\vec{m}|\vec{x},\theta) & \text{MAR} \\ P(\vec{m}|\vec{x},\vec{h},\theta) & \text{not-MAR (NMAR)} \end{cases} . \quad (28.1)$$

For doing imputation via EM, we connect node $\underline{\vec{m}}$ as shown in the middle bnet (called MAR) of Fig.28.1.

For the example of Table 28.1, we have variables $\underline{\vec{m}}, \underline{\vec{x}}$ and $\underline{\vec{h}}$ whose values range over the following sets:

$\underline{\vec{x}} = (\underline{\vec{x}}_0, \underline{\vec{x}}_1, \underline{\vec{x}}_2)$
$\underline{\vec{h}} = (\underline{\vec{h}}_0)$
$\underline{h}_0[\sigma] \in \{0, 1\}$,
$\underline{x}_i[\sigma] \in \{0, 1\}$ for $i = 0, 1, 2$,
$\underline{m}_i[\sigma] \in \{0, 1\}$ for $i = 0, 1, 2$.

For concreteness, we will assume that the Markov chain $\underline{m}[\sigma] \leftarrow \underline{x}[\sigma] \leftarrow \underline{h}[\sigma]$ has a finer grained DAG structure given by Fig.28.3. where we will omit the dashed arrows. If one doesn't want to assume that the data can be fitted well by the bnet of Fig.28.3 without the dashed arrows, one can include those arrows too, at the expense of more unknown parameters (i.e., degrees of freedom)

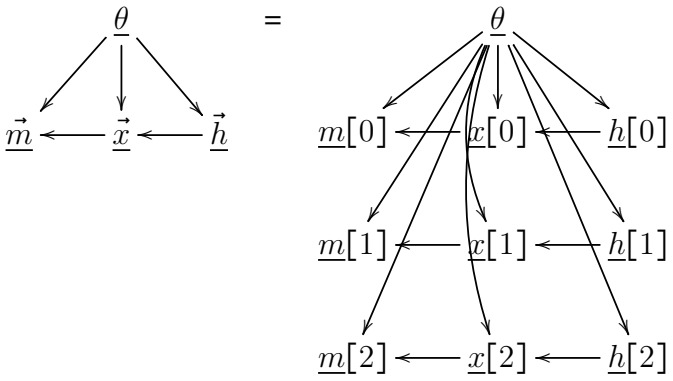


Figure 28.2: MAR bnet with $nsam = 3$.

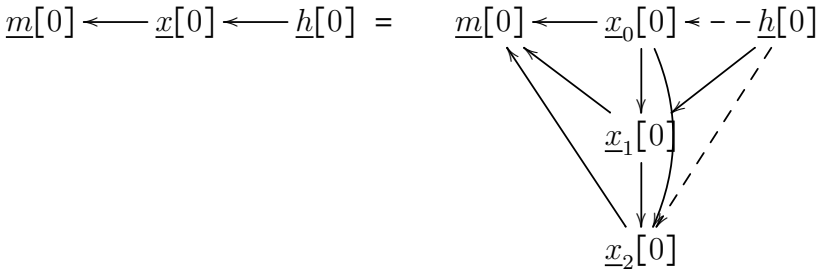


Figure 28.3: Our example for imputation via EM assumes this bnet between nodes $\underline{m}[\sigma], \underline{x}[\sigma], \underline{h}[\sigma]$.

to be lumped into θ. We will parmaterize the TPMs corresponding to Fig.28.3 using a Categorical Distribution for each column of the TPMs. We will thus assume that the bnet of Fig.28.3 has the following TPMs, printed in blue.

$$P(h_0^{[\sigma]}|\theta) = \begin{array}{c|c} & \\ \hline & 1-\theta_0 \\ 1 & \theta_0 \end{array} \tag{28.2}$$

$$P(x_0^{[\sigma]}|\theta) = \begin{array}{c|c} & \\ \hline 0 & 1-\theta_1 \\ 1 & \theta_1 \end{array} \tag{28.3}$$

$$P(x_1^{[\sigma]} \mid x_0^{[\sigma]}, h^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1-\theta_2 & 1-\theta_3 & 1-\theta_4 & 1-\theta_5 \\ 1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{array} \tag{28.4}$$

$$P(x_2^{[\sigma]} \mid x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1-\theta_6 & 1-\theta_7 & 1-\theta_8 & 1-\theta_9 \\ 1 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \end{array} \tag{28.5}$$

$$P(m^{[\sigma]}|x^{[\sigma]}, \theta) = \frac{1}{nsam} P((x_i)_{\forall i \ni m_i=1} \mid (x_i)_{\forall i \ni m_i=0}, \theta) \tag{28.6}$$

Eq.(28.6) can be illustrated as follows. In Table 28.2, we added a $P(m)$ column to Table 28.1.

	h_0	x_0	x_1	x_2	m	$P(m)$
1	NA	0	1	1	(0,0,0)	$\frac{1}{nsam}$
2	NA	0	0	0	(0,0,0)	$\frac{1}{nsam}$
3	NA	1	1	0	(0,0,0)	$\frac{1}{nsam}$
4	NA	0 0 1 1	1	0 1 0 1	(1,0,1)	$\frac{1}{nsam}P(x_0=0, x_2=0 \mid x_1=1, \theta)$ $\frac{1}{nsam}P(x_0=0, x_2=1 \mid x_1=1, \theta)$ $\frac{1}{nsam}P(x_0=1, x_2=0 \mid x_1=1, \theta)$ $\frac{1}{nsam}P(x_0=1, x_2=1 \mid x_1=1, \theta)$
5	NA	0	0 1	1	(0,1,0)	$\frac{1}{nsam}P(x_1=0 \mid x_0=0, x_2=1, \theta)$ $\frac{1}{nsam}P(x_1=1 \mid x_0=0, x_2=1, \theta)$
6	NA	0	0	1	(0,0,0)	$\frac{1}{nsam}$

Table 28.2: $P(m)$ column added to Table 28.1. Note that $\sum_m P(m) = 1$.

$$\theta = (\theta_i)_{i=0,1,\ldots,9} \tag{28.7}$$

$$P(m^{[\sigma]}, x^{[\sigma]}, h^{[\sigma]}|\theta) = P(m^{[\sigma]}|x^{[\sigma]}, \theta) P(x^{[\sigma]}|h^{[\sigma]}, \theta) P(h^{[\sigma]}|\theta) \tag{28.8}$$

$$P(x^{[\sigma]}|h^{[\sigma]}, \theta) = P(x_2^{[\sigma]}|x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) P(x_1^{[\sigma]}|x_0^{[\sigma]}, h^{[\sigma]}, \theta) P(x_0^{[\sigma]}|\theta) \tag{28.9}$$

$$P(x_1^{[\sigma]}|x_0^{[\sigma]}, \theta) = \sum_h P(x_1^{[\sigma]}|x_0^{[\sigma]}, h^{[\sigma]}, \theta) P(h^{[\sigma]}|\theta) \tag{28.10}$$

$$P(x^{[\sigma]}|\theta) = P(x_2^{[\sigma]}|x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) P(x_1^{[\sigma]}|x_0^{[\sigma]}, \theta) P(x_0^{[\sigma]}|\theta) \tag{28.11}$$

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= \sum_{\vec{m},\vec{h}} P(\vec{m},\vec{h} \mid \vec{x},\theta^{(t)}) \ln P(\vec{m},\vec{x},\vec{h}|\theta) && (28.12)\\
&= \sum_{\vec{m},\vec{h}} \left[\prod_{\sigma} P(m^{[\sigma]},h^{[\sigma]} \mid x^{[\sigma]},\theta^{(t)}) \right] \ln \left[\prod_{\sigma} P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) \right] && (28.13)\\
&= \sum_{\sigma} \sum_{m^{[\sigma]},h^{[\sigma]}} P(m^{[\sigma]},h^{[\sigma]} \mid x^{[\sigma]},\theta^{(t)}) \ln P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) && (28.14)\\
&= \sum_{\sigma} \sum_{m^{[\sigma]},h^{[\sigma]}} \frac{P(m^{[\sigma]},h^{[\sigma]},x^{[\sigma]} \mid \theta^{(t)})}{P(x^{[\sigma]} \mid \theta^{(t)})} \ln P(m^{[\sigma]},x^{[\sigma]},h^{[\sigma]}|\theta) && (28.15)
\end{aligned}
$$

Once you find optimal parameters θ^* by recursing this $Q(\theta|\theta^{(t)})$, you can evaluate numerically the $P(m)$ column of Table 28.2. In Table 28.2, out of the 4 sub-rows for row 4, choose the one with the highest probability. Similarly, out of the 2 sub-rows for row 5, choose the one with the highest probability.

Imputation via MCMC

A simple and popular way to do inputation via MCMC is described in Ref.[27]. It goes as follows.

Let

$$\underline{H}[\sigma] = (\underline{h}[\sigma], \underline{m}[\sigma]) \tag{28.16}$$

for $\sigma = 0, 1, \ldots, nsam-1$. Initialize $\theta^{(0)}$ to a random value within the allowed ranges. Do the following 2 steps, for $t = 0, 1, \ldots, T-1$, where T is large enough that $\theta^{(t)}$ has reached a steady value that is independent of $\theta^{(0)}$. To do the sampling, use a standard sampling technique such as Gibbs sampling.

- **STEP 1:** For $\sigma = 0, 1, \ldots, nsam-1$, find a sample

$$(H^{[\sigma]})^{(t+1)} \sim P(H^{[\sigma]}|x^{[\sigma]},\theta^{(t)}) \; . \tag{28.17a}$$

- **STEP 2:** Find a sample

$$\theta^{(t+1)} \sim P^{(t+1)}(\theta) \tag{28.17b}$$

where

$$
\begin{aligned}
P^{(t+1)}(\theta) &= \mathcal{N}(!\theta) P(\vec{x},\vec{H}^{(t+1)}|\theta) && (28.17c)\\
&= \mathcal{N}(!\theta) \prod_{\sigma} P(x^{[\sigma]},(H^{[\sigma]})^{(t+1)}|\theta) \; . && (28.17d)
\end{aligned}
$$

Fig.28.4 illustrates this two step recursive process using a bnet.

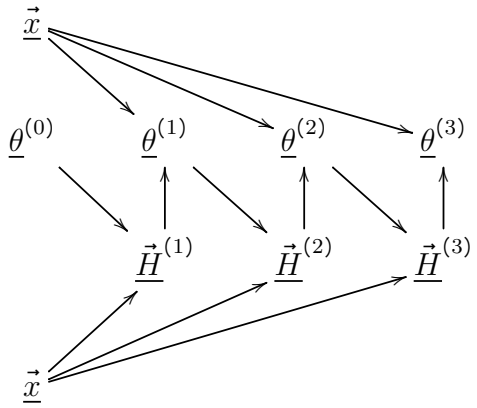


Figure 28.4: bnet illustrating Eqs.(28.17) for doing imputation via MCMC. The **same** node $\underline{\vec{x}}$ appears twice to make the graph clearer.

Multiple Imputations

Multiple imputations means calculating θ^* (i.e., the optimum θ) and the concomitant dataset $\vec{x}^*, \vec{H}^*$, via any method (such as EM or MCMC), a large number of times, starting from different, randomly chosen $\theta^{(0)}$ initial parameters. Then calculating the average and the variance of $\theta^*, \vec{x}^*, \vec{H}^*$ and functions thereof.

Chapter 29

Monty Hall Problem

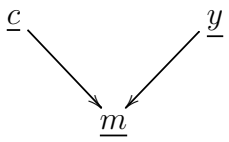


Figure 29.1: Monty Hall Problem.

Mr. Monty Hall, host of the game show "Lets Make a Deal", hides a car behind one of three doors and a goat behind each of the other two. The contestant picks Door No. 1, but before opening it, Mr. Hall opens Door No. 2 to reveal a goat. Should the contestant stick with No. 1 or switch to No. 3?

The Monty Hall problem can be modeled by the bnet Fig.29.1, where

- $\underline{c}$= the door behind which the car actually is.
- $\underline{y}$= the door opened by you (the contestant), on your first selection.
- $\underline{m}$= the door opened by Monty (game host)

We label the doors 1,2,3 so $S_{\underline{c}} = S_{\underline{y}} = S_{\underline{m}} = \{1, 2, 3\}$.
Node matrices printed in blue:

$$P(c) = \frac{1}{3} \text{ for all } c \tag{29.1}$$

$$P(y) = \frac{1}{3} \text{ for all } y \tag{29.2}$$

$$P(m|c, y) = \mathbb{1}(m \neq c)\left[\frac{1}{2}\mathbb{1}(y = c) + \mathbb{1}(y \neq c)\mathbb{1}(m \neq y)\right] \tag{29.3}$$

It's easy to show that the above node probabilities imply that

$$P(c = 1|m = 2, y = 1) = \frac{1}{3} \tag{29.4}$$

$$P(c = 3|m = 2, y = 1) = \frac{2}{3} \tag{29.5}$$

So you are twice as likely to win if you switch your final selection to be the door which is neither your first choice nor Monty's choice.

The way I justify this to myself is: Monty gives you a piece of information. If you don't switch your choice, you are wasting that info, whereas if you switch, you are using the info.

Chapter 30

Naive Bayes

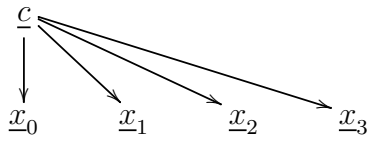


Figure 30.1: bnet for Naive Bayes with 4 features

Class node $\underline{c} \in S_{\underline{c}}$. $|S_{\underline{c}}| = n_{\underline{c}}$= number of classes.
Feature nodes $\underline{x}_i \in S_{\underline{x}_i}$ for $i = 0, 1, 2, \ldots, F-1$. F=number of features.
Define

$$x. = [x_0, x_1, \ldots, x_{F-1}] \; . \tag{30.1}$$

For the bnet of Fig.30.1,

$$P(c, x.) = P(c) \prod_{i=0}^{F-1} P(x_i|c) \; . \tag{30.2}$$

Given $x.$ values, find most likely class $c \in S_{\underline{c}}$.
Maximum a Posteriori (MAP) estimate:

$$c^* = \operatorname*{argmax}_c P(c|x.) \tag{30.3}$$

$$= \operatorname*{argmax}_c \frac{P(c, x.)}{P(x.)} \tag{30.4}$$

$$= \operatorname*{argmax}_c P(c, x.) \; . \tag{30.5}$$

Chapter 31

Neural Networks

In this chapter, we discuss Neural Networks (NNs) of the feedforward kind, which is the most popular kind. In their plain, vanilla form, NNs only have deterministic nodes. But the nodes of a bnet can be deterministic too, because the TPM of a node can reduce to a delta function. Hence, NNs should be expressible as bnets. We will confirm this in this chapter.

Henceforth in this chapter, if we replace an index of an indexed quantity by a dot, it will mean the collection of the indexed quantity for all values of that index. For example, $x.$ will mean the array of x_i for all i.

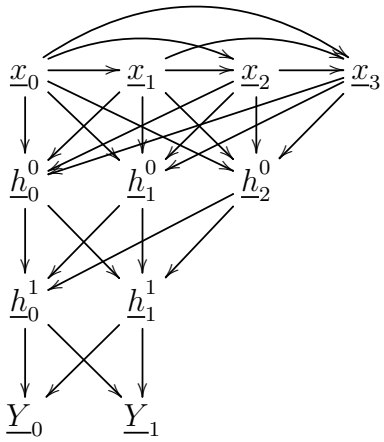


Figure 31.1: Neural Network (feed forward) with 4 layers: input layer $\underline{x}.$, 2 hidden layers $\underline{h}^0.$, $\underline{h}^1.$ and output layer $\underline{Y}$.

Consider Fig.31.1.

$\underline{x}_i \in \{0,1\}$ for $i = 0,1,2,\ldots,nx-1$ is the **input layer**.

$\underline{h}^\lambda_i \in \mathbb{R}$ for $i = 0,1,2,\ldots,nh(\lambda)-1$ is the **λ-th hidden layer**. $\lambda = 0,1,2,\ldots,\Lambda-2$. A NN is said to be **deep** if $\Lambda > 2$; i.e., if it has more than one hidden layer.

$\underline{Y}_i \in \mathbb{R}$ for $i = 0,1,2,\ldots,ny-1$ is the **output layer**. We use a upper case y here because in the training phase, we will use pairs $(x.[\sigma], y.[\sigma])$ where $y_i[\sigma] \in \{0,1\}$ for $i = 0,1,\ldots,ny-1$.

$Y = \hat{y}$ is an estimate of y. Note that lower case y is either 0 or 1, but upper case y may be any real. Often, the activation functions are chosen so that $Y \in [0, 1]$.

The number of nodes in each layer and the number of layers are arbitrary. Fig.31.1 is fully connected (aka dense), meaning that every node of a layer is impinged arrow coming from every node of the preceding layer. Later on in this chapter, we will discuss non-dense layers.

Let $w^{\lambda}_{i|j}, b^{\lambda}_{i} \in \mathbb{R}$ be given, for $i \in [0, nh(\lambda))_{\mathbb{Z}}$, $j \in [0, nh(\lambda - 1))_{\mathbb{Z}}$, and $\lambda \in [0, \Lambda)_{\mathbb{Z}}$.

These are the TPMs, printed in blue, for the nodes of the bnet Fig.31.1:

$$P(x_i \mid x_{i-1}, x_{i-1}, \ldots, x_0) = \text{given} \tag{31.1}$$

$$P(h^{\lambda}_{i} \mid h^{\lambda-1}_{\cdot}) = \delta\left(h^{\lambda}_{i}, \mathcal{A}^{\lambda}_{i}(\sum_{j} w^{\lambda}_{i|j} h^{\lambda-1}_{j} + b^{\lambda}_{i})\right), \tag{31.2}$$

where $P(h^0_i | h^{-1}) = P(h^0_i | x)$.

$$P(Y_i \mid h^{\Lambda-2}_{\cdot}) = \delta\left(Y_i, \mathcal{A}^{\Lambda-1}_{i}(\sum_{j} w^{\Lambda-1}_{i|j} h^{\Lambda-2}_{j} + b^{\Lambda-1}_{i})\right). \tag{31.3}$$

Activation Functions $\mathcal{A}^{\lambda}_{i} : \mathbb{R} \to \mathbb{R}$

Activation functions must be nonlinear.

- **Step function (Perceptron)**

$$\mathcal{A}(x) = \mathbb{1}(x > 0) \tag{31.4}$$

Zero for $x \leq 0$, one for $x > 0$.

- **Sigmoid function**

$$\mathcal{A}(x) = \frac{1}{1 + e^{-x}} = \text{sig}(x) \tag{31.5}$$

Smooth, monotonically increasing function. $\text{sig}(-\infty) = 0$,$\text{sig}(0) = 0.5$, $\text{sig}(\infty) = 1$.

$$\begin{aligned} \text{sig}(x) + \text{sig}(-x) &= \frac{1}{1 + e^{-x}} + \frac{1}{1 + e^{x}} && (31.6)\\ &= \frac{2 + e^{x} + e^{-x}}{2 + e^{x} + e^{-x}} && (31.7)\\ &= 1 && (31.8) \end{aligned}$$

- **Hyperbolic tangent**

$$\mathcal{A}(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{31.9}$$

Smooth, monotonically increasing function. $\tanh(-\infty) = -1$,$\tanh(0) = 0$, $\tanh(\infty) = 1$.

Odd function:

$$\tanh(-x) = -\tanh(x) \tag{31.10}$$

Whereas $\text{sig}(x) \in [0, 1]$, $\tanh(x) \in [-1, 1]$.

- **ReLU (Rectified Linear Unit)**

$$\mathcal{A}(x) = x\mathbb{1}(x > 0) = \max(0, x) \ . \tag{31.11}$$

Compare this to the step function.

- **Swish**

$$\mathcal{A}(x) = x \ \text{sig}(x) \tag{31.12}$$

- **Softmax**

$$\mathcal{A}(x_i|x.) = \frac{e^{x_i}}{\sum_i e^{x_i}} \tag{31.13}$$

It's called softmax because if we approximate the exponentials, both in the numerator and denominator of Eq.(31.13), by the largest one, we get

$$\mathcal{A}(x_i|x.) \approx \mathbb{1}(x_i = \max_k x_k) \ . \tag{31.14}$$

The softmax definition implies that the bnet nodes within a softmax layer are fully connected by arrows to form a "clique".

For 2 nodes x_0, x_1,

$$\mathcal{A}(x_0|x.) = \frac{e^{x_0}}{e^{x_0} + e^{x_1}} \tag{31.15}$$

$$= \text{sig}(x_0 - x_1) \ , \tag{31.16}$$

$$\mathcal{A}(x_1|x.) = \text{sig}(x_1 - x_0) \ . \tag{31.17}$$

Weight optimization via supervised training and gradient descent

The bnet of Fig.31.1 is used for classification of a single data point $x.$. It assumes that the weights $w^{\lambda}_{i|j}, b^{\lambda}_{i}$ are given.

To find the optimum weights via supervised training and gradient descent, one uses the bnet Fig.31.2.

In Fig.31.2, the nodes in Fig.31.1 become sampling space vectors. For example, $\underline{x}.$ becomes $\underline{\vec{x}}.$, where the components of $\underline{\vec{x}}.$ in sampling space are $\underline{x}.[\sigma] \in \{0,1\}^{nx}$ for $\sigma = 0, 1, \ldots, nsam(\vec{x}) - 1$.

$nsam(\vec{x})$ is the number of samples used to calculate the gradient during each **stage (aka iteration)** of Fig.31.2. We will also refer to $nsam(\vec{x})$ as the **mini-batch size**. A **mini-batch** is a subset of the training data set.

To train a bnet with a data set (d-set), the standard procedure is to split the d-set into 3 parts:

1. **training d-set**,
2. **testing1 d-set**, for tuning of hyperparameters like $nsam(\vec{x})$, Λ, and $nunh(i)$ for each i.
3. **testing2 d-set**, for measuring how well the model tuned with the testing1 d-set performs.

The training d-set is itself split into mini-batches. An **epoch** is a pass through all the training d-set.

Define

$$W^{\lambda}_{i|j} = [w^{\lambda}_{i|j}, b^{\lambda}_{i}] \, . \tag{31.18}$$

These are the TPMs, printed in blue, for the nodes of the bnet Fig.31.2:

$$P(x.[\sigma]) = \text{given} \, . \tag{31.19}$$

$$P(y.[\sigma] \,|\, x.[\sigma]) = \text{given} \, . \tag{31.20}$$

$$P(h^{\lambda}_{i}[\sigma] \,|\, h^{\lambda-1}_{\cdot}[\sigma]) = \delta\left(h^{\lambda}_{i}[\sigma], \mathcal{A}^{\lambda}_{i}(\sum_{j} w^{\lambda}_{i|j} h^{\lambda-1}_{j}[\sigma] + b^{\lambda}_{i})\right) \tag{31.21}$$

$$P(Y_{i}[\sigma] \,|\, h^{\Lambda-2}_{\cdot}[\sigma]) = \delta\left(Y_{i}[\sigma], \mathcal{A}^{\Lambda-1}_{i}(\sum_{j} w^{\Lambda-1}_{i|j} h^{\Lambda-2}_{j}[\sigma] + b^{\Lambda-1}_{i})\right) \tag{31.22}$$

$$P(W^{\cdot}_{\cdot|\cdot}) = \text{given} \tag{31.23}$$

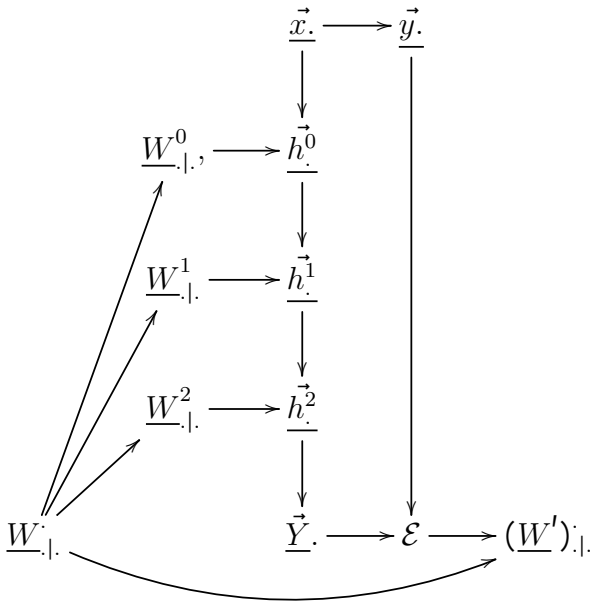


Figure 31.2: bnet for finding optimum weights of the bnet Fig.31.1 via supervised training and gradient descent.

The first time it is used, $W^{\cdot}_{\cdot|\cdot}$ is arbitrary. After the first time, it is determined by previous stage.

$$P(W^{\lambda}_{\cdot|\cdot}|W^{\cdot}_{\cdot|\cdot}) = \delta(W^{\lambda}_{\cdot|\cdot}, (W^{\cdot}_{\cdot|\cdot})^{\lambda}) \tag{31.24}$$

$$P(\mathcal{E}|\vec{y}_{\cdot}, \vec{Y}_{\cdot}) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \sum_{i} d(y_i[\sigma], Y_i[\sigma]) \ , \tag{31.25}$$

where

$$d(y, Y) = |y - Y|^2 \ . \tag{31.26}$$

If $y, Y \in [0, 1]$, one can use this instead

$$d(y, Y) = XE(y \to Y) = -y \ln Y - (1 - y) \ln(1 - Y) \ . \tag{31.27}$$

$$P((W')^{\lambda}_{i|j}|\mathcal{E}, W^{\cdot}_{\cdot|\cdot}) = \delta((W')^{\lambda}_{i|j}, W^{\lambda}_{i|j} - \eta \partial_{W^{\lambda}_{i|j}} \mathcal{E}) \tag{31.28}$$

$\eta > 0$ is called the learning rate. This method of minimizing the error $\mathcal{E}$ is called gradient descent. $W' - W = \Delta W = -\eta \partial_W \mathcal{E}$ so $\Delta \mathcal{E} = \frac{-1}{\eta} (\Delta W)^2 < 0$.

Non-dense layers

The TPM for a non-dense layer is of the form:

$$P(h_i^\lambda[\sigma] \mid h_\cdot^{\lambda-1}[\sigma]) = \delta(h_i^\lambda[\sigma], H_i^\lambda[\sigma]) \ , \tag{31.29}$$

where $H_i^\lambda[\sigma]$ will be specified below for each type of non-dense layer.

- **Dropout Layer**

  The dropout layer was invented in Ref.[26]. To dropout nodes from a fixed layer λ: For all i of layer λ, define a new node $\underline{r}_i^\lambda$ with an arrow $\underline{r}_i^\lambda \to \underline{h}_i^\lambda$. For $r \in \{0,1\}$, and some $p \in (0,1)$, define

  $$P(r_i^\lambda = r) = [p]^r[1-p]^{1-r} \text{ (Bernouilli dist.)} \ . \tag{31.30}$$

  Now one has

  $$P(h_i^\lambda[\sigma] \mid h_\cdot^{\lambda-1}[\sigma], r_i^\lambda) = \delta(h_i^\lambda[\sigma], H_i^\lambda[\sigma]) \ , \tag{31.31}$$

  where

  $$H_i^\lambda[\sigma] = \mathcal{A}_i^\lambda(r_i^\lambda \sum_j w_{i|j}^\lambda h_j^{\lambda-1}[\sigma] + b_i^\lambda) \ . \tag{31.32}$$

  This reduces ovefitting. Overfitting might occur if the weights follow too closely several similar minibatches. This dropout procedure adds a random component to each minibatch making groups of similar minibatches less likely.

  The random $\underline{r}_i^\lambda$ nodes that induce dropout are only used in the training bnet Fig.31.2, not in the classification bnet Fig.31.1. We prefer to remove the $\underline{r}_i^\lambda$ stochasticity from classification and for Fig.31.1 to act as an average over sampling space of Fig.31.2. Therefore, if weights $w_{i|j}^\lambda$ are obtained for a dropout layer λ in Fig.31.2, then that layer is used in Fig.31.1 with no $\underline{r}_i^\lambda$ nodes but with weights $\langle r_i^\lambda \rangle w_{i|j}^\lambda = p w_{i|j}^\lambda$.

  Note that dropout adds non-deterministic nodes to a NN, which in their vanilla form only have deterministic nodes.

- **Convolutional Layer**

  - 1-dim
    Filter function $\mathcal{F} : \{0, 1, \ldots, nf-1\} \to \mathbb{R}$.
    σ=stride length

For $i \in \{0, 1, \ldots, nh(\lambda) - 1\}$, let

$$H_i^\lambda[\sigma] = \sum_{j=0}^{nf-1} h_{j+i\sigma}^{\lambda-1}[\sigma]\mathcal{F}(j) \ . \tag{31.33}$$

For the indices not to go out of bounds in Eq.(31.33), we must have

$$nh(\lambda - 1) - 1 = nf - 1 + (nh(\lambda) - 1)\sigma \tag{31.34}$$

so

$$nh(\lambda) = \frac{1}{\sigma}[nh(\lambda - 1) - nf] + 1 \ . \tag{31.35}$$

- 2-dim
  $h_i^\lambda[\sigma]$ becomes $h_{(i,j)}^\lambda[\sigma]$. Do 1-dim convolution along both i and j axes.

- **Pooling Layers (MaxPool, AvgPool)**

  Here each node i of layer λ is impinged by arrows from a subset $Pool(i)$ of the set of all nodes of the previous layer $\lambda - 1$. Partition set $\{0, , 1, \ldots, nh(\lambda - 1) - 1\}$ into $nh(\lambda)$ mutually disjoint, nonempty sets called $Pool(i)$, where $i \in \{0, 1, \ldots, nh(\lambda) - 1\}$.

  - AvgPool
  $$H_i^\lambda[\sigma] = \frac{1}{size(Pool(i))} \sum_{j \in Pool(i)} h_j^{\lambda-1}[\sigma] \tag{31.36}$$

  - MaxPool
  $$H_i^\lambda[\sigma] = \max_{j \in Pool(i)} h_j^{\lambda-1}[\sigma] \tag{31.37}$$

Autoencoder NN

If the sequence

$$nx, nh(0), nh(1), \ldots, nh(\Lambda - 2), ny \tag{31.38}$$

first decreases monotonically up to layer λ_{min}, then increases monotonically until $ny = nx$, then the NN is called an **autoencoder NN**. Autoencoders are useful for unsupervised learning and feature reduction. In this case, Y estimates x. The layers before layer λ_{min} are called the **encoder**, and those after λ_{min} are called the **decoder**. Layer λ_{min} is called the **code**.

Chapter 32

Noisy-OR gate

The Noisy-OR gate was first proposed by Judea Pearl in his 1988 book Ref.[18].

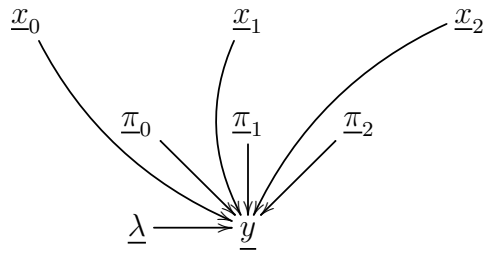


Figure 32.1: Noisy-OR gate $\underline{y} \in \{0,1\}$ with $n = 3$, Boolean inputs $(\underline{x}_i)_{i=0,1,2}$ and parameters $\underline{\lambda}, (\underline{\pi})_{i=0,1,2}$.

Let
$\underline{\lambda} \in [0,1]$ =**gate leakage**.
$y \in \{0,1\}$ = gate output
$\underline{x}^n = (\underline{x}_i)_{i=0,1,\ldots,n-1}$, where $\underline{x}_i \in \{0,1\}$ are gate inputs.
$\underline{\pi}^n = (\underline{\pi}_i)_{i=0,1,\ldots,n-1}$, where $\underline{\pi}_i \in [0,1]$ are gate parameters.
The TPM, printed in blue, for the Noisy-OR gate $\underline{y}$ shown in Fig.32.1, is

$$P(y = 1|x^n, \lambda, \pi^n) = 1 - (1 - \lambda) \prod_i [1 - \pi_i x_i] \tag{32.1}$$

$$P(y = 0|x^n, \lambda, \pi^n) = 1 - P(y = 1|x^n, \lambda, \pi^n) \tag{32.2}$$

Note that if $\lambda = 0$ and $\pi_i = 1$ for all i, then this becomes a deterministic OR-gate. Indeed,

$$P(y = 1|x^n, \lambda = 0, \pi^n = 1^n) = 1 - \prod_i [1 - x_i] = \vee_{i=0}^{n-1} x_i \ , \tag{32.3}$$

so

$$P(y|x^n, \lambda = 0, \pi^n = 1^n) = \delta(y, \vee_{i=0}^{n-1} x_i) \ . \tag{32.4}$$

3 ways to interpret the parameters π_i:

1. Note that if $\lambda = 0$ and x^n is one hot (i.e., $x^n = e_i^n$, where e_i^n is the vector with all components zero except for the i-th component which equals 1), then

$$P(y = 1|x^n = e_i^n, \lambda = 0, \pi^n) = 1 - [1 - \pi_i] = \pi_i \ . \tag{32.5}$$

   This gives an interpretation to the parameters π_i.

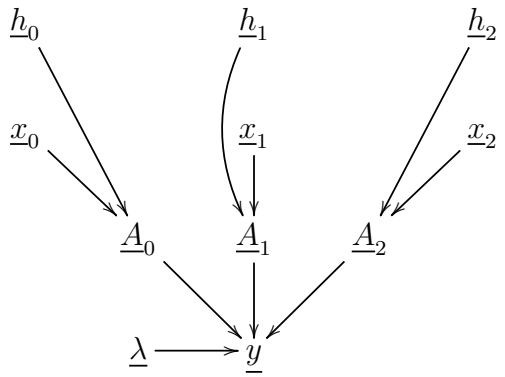


Figure 32.2: Fig.32.1 after replacing parameters $(\underline{\pi}_i)_{i=0,1,2}$ by hidden nodes $(\underline{h}_i)_{i=0,1,2}$.

2. Another way of interpreting the parameters π_i is to associate each of them with a hidden variable $\underline{h}_i \in \{0, 1\}$ whose average equals π_i. More precisely, consider Fig.32.2.

   Let $\underline{x}_i, \underline{h}_i, \underline{A}_i, \underline{y} \in \{0, 1\}$.

   The TPMs, printed in blue, for the nodes of the bnet Fig.32.2, are as follows:

$$P(h_i) = \pi_i \delta(h_i, 1) + (1 - \pi_i)\delta(h_i, 0) \tag{32.6}$$

$$P(A_i|h_i, x_i) = \delta(A_i, h_i \wedge x_i) = \delta(A_i, h_i x_i) \tag{32.7}$$

$$\begin{aligned} P(y = 1|A^n) &= 1 - (1 - \lambda) \wedge_{i=0}^{n-1} \overline{A}_i & (32.8) \\ &= 1 - (1 - \lambda) \prod_i (1 - A_i) & (32.9) \end{aligned}$$

$$P(y=0|A^n) = 1 - P(y=1|A^n) \tag{32.10}$$

Note that

$$P(y=1|x^n,\lambda) = \sum_{h^n}\sum_{A^n}\left[1-(1-\lambda)\prod_i(1-A_i)\right][\prod_i \delta(A_i,h_ix_i)]P(h^n) \tag{32.11}$$

$$= E_{\underline{h}^n}\left[[1-(1-\lambda)\prod_i(1-h_ix_i)\right] . \tag{32.12}$$

But

$$E_{\underline{h}_i}[h_ix_i] = \sum_{h_i=0,1} P(h_i)h_ix_i = \pi_ix_i \tag{32.13}$$

so

$$P(y=1|x^n,\lambda) = 1-(1-\lambda)\prod_i(1-\pi_ix_i) . \tag{32.14}$$

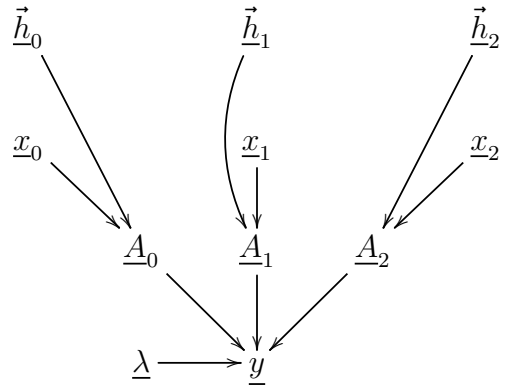


Figure 32.3: Fig.32.2 after replacing the hidden nodes $(\underline{h}_i)_{i=0,1,2}$ by vectors of samples $(\vec{\underline{h}}_i)_{i=0,1,2}$.

3. Another way to interpret the parameters π_i is to associate each of them with a vector of samples $\vec{\underline{h}}_i$ whose average is π_i. More precisely, consider Fig.32.3.

   Suppose $\underline{h}_i \in \{0,1\}$ and define

   $$P_{\underline{h}_i}(h_i) = \pi_i\delta(h_i,1) + (1-\pi_i)\delta(h_i,0) . \tag{32.15}$$

   Suppose $\vec{\underline{h}}_i = (\underline{h}_i[\sigma])_{s=0,1,\ldots,nsam-1}$ and the Boolean samples $\underline{h}_i[\sigma] \in \{0,1\}$ are i.i.d. with $\underline{h}_i[\sigma] \sim P_{\underline{h}_i}$ for all σ.

Note that for each i, an estimate $\hat{P}_{\underline{h}_i}(h_i)$ of $P_{\underline{h}_i}(h_i)$ can be obtained from the vector of samples $\vec{\underline{h}}_i$ as follows:

$$\hat{P}_{\underline{h}_i}(h_i) = \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \mathbb{1}(h_i[\sigma] = h_i) \ . \tag{32.16}$$

Let $\underline{x}_i, \underline{h}_i[\sigma], \underline{A}_i, \underline{y} \in \{0, 1\}$.
The TPMs, printed in blue, for the nodes of the bnet Fig.32.3, are as follows:

$$P(\vec{h}_i) = \prod_{\sigma=0}^{nsam-1} P_{\underline{h}}(h_i[\sigma]) \tag{32.17}$$

$$\begin{aligned} P(A_i \mid \vec{h}_i, x_i) &= \delta(A_i, \frac{1}{nsam} \sum_{\sigma} h_i[\sigma] \wedge x_i) && (32.18) \\ &= \delta(A_i, \pi_i x_i) && (32.19) \end{aligned}$$

$$\begin{aligned} P(y = 1|A^n) &= 1 - (1 - \lambda) \wedge_{i=0}^{n-1} \overline{A}_i && (32.20) \\ &= 1 - (1 - \lambda) \prod_i (1 - A_i) && (32.21) \end{aligned}$$

$$P(y = 0|A^n) = 1 - P(y = 1|A^n) \tag{32.22}$$

Note that

$$\begin{aligned} P(y = 1|x^n, \lambda, \vec{h}^n) &= \sum_{A^n} \left[1 - (1 - \lambda) \prod_i (1 - A_i) \right] \prod_i \delta(A_i, \pi_i x_i) && (32.23) \\ &= 1 - (1 - \lambda) \prod_i (1 - \pi_i x_i) \ . && (32.24) \end{aligned}$$

Chapter 33

Non-negative Matrix Factorization

Based on Ref.[58].

Given matrix V, factor it into product of two matrices

$$V = WH \ , \tag{33.1}$$

where all 3 matrices have non-negative entries.
$V \in \mathbb{R}_{\geq 0}^{nv \times na}$: visible info matrix
$W \in \mathbb{R}_{\geq 0}^{nv \times nh}$: weight info matrix
$H \in \mathbb{R}_{\geq 0}^{nh \times na}$: hidden info matrix
Usually, $nv > nh < na$ so compression of information (aka dimensional reduction, clustering)

B net interpretation: Express node $\underline{v}$ as a chain of two nodes.

$$\underline{v} \longleftarrow \underline{a} \qquad = \qquad \underline{w} \longleftarrow \underline{h} \longleftarrow \underline{a}$$

Figure 33.1: B net interpretation of non-negative matrix factorization.

Node TPMs, printed in blue, for Fig.33.1.

$$P(\underline{v} = w|a) = \frac{V_{w,a}}{\sum_w V_{w,a}} \tag{33.2}$$

$$P(w|h) = \frac{W_{w,h}}{\sum_w W_{w,h}} \tag{33.3}$$

$$P(h|a) = \frac{\sum_w W_{w,h}}{\sum_w V_{w,a}} H_{h,a} \tag{33.4}$$

Simplest recursive algorithm:

Initialize: Choose nh. Choose $W^{(0)}$ and $H^{(0)}$ that have non-negative entries.

Update: For $n = 0, 1, \ldots,$ do

$$H_{i,j}^{(n+1)} \leftarrow H_{i,j}^{(n)} \frac{[(W^{(n)})^T V]_{i,j}}{[(W^{(n)})^T \underbrace{W^{(n)} H^{(n)}}_{\approx V}]_{i,j}} \tag{33.5}$$

and

$$W_{i,j}^{(n+1)} \leftarrow W_{i,j}^{(n)} \frac{[V(H^{(n+1)})^T]_{i,j}}{[\underbrace{W^{(n)} H^{(n+1)}}_{\approx V}(H^{(n+1)})^T]_{i,j}} \ . \tag{33.6}$$

After each step, record error defined by

$$\mathcal{E}^{(n)} = \| \ V - W^{(n)} H^{(n)} \ \|_2 \ \ . \tag{33.7}$$

Using 2-norm, aka Frobenius matrix norm. Continue until reach acceptable error.

Can also use Kullback-Lieber divergence for error:

$$\mathcal{E} = \sum_a P(a) D_{KL}(P(\underline{v} = w|a) \ \| \ \sum_h P(w|h) P(h|a)) \ , \tag{33.8}$$

for some arbitrary choice of prior $P(a)$. For example, can choose $P(a)$ uniform.

Chapter 34

Observational Equivalence of DAGs

This chapter is based on Chapter 1 of Ref.[19] and on a blog post by Bruno Gonçalves (Ref.[4]).

A probability distribution P **is compatible with a DAG** G if P and G have the same random variables, and they can be combined to form a bnet without contradictions; i.e., one can calculate all the TPMs from P and multiply them together to obtain P again. Let

$$\mathcal{P}(G) = \{P : P \text{ is compatible with } G\} . \tag{34.1}$$

Two DAGs G **and** G' **are observationally equivalent (OE)** if $\mathcal{P}(G) = \mathcal{P}(G')$. Hence, any total probability distribution that is compatible with one of them is compatible with the other. For example, $\underline{a} \rightarrow \underline{b}$ and $\underline{a} \leftarrow \underline{b}$ are OE because

$$P(a|b)P(b) = P(a,b) = P(b|a)P(a) . \tag{34.2}$$

We'll say two bnets are OE if their DAGs are OE.

Two DAGs G and G' are **d-separation equivalent** if $DS(G) = DS(G')$. See Chapter 12 for definition of $DS(G)$.

Claim 21 *Two DAGs are OE iff their DAGs are d-separation equivalent.*

The **skeleton** of a DAG is its undelying undirected graph.

A **v-structure** in a DAG consists of two arrows converging to a node and such that their tails are not connected by a third arrow. Fig.34.1 shows in red all the v-structures of a particular DAG.

Claim 22 *Observational Equivalence Theorem (by Verma and Pearl, 1990)*
Two DAGs are OE iff they have the same skeletons and the same v-structures.

Example

The 3 DAGs in Fig.34.2 are OE. They form an equivalence class of OE DAGs that represent the same probability distribution. This equivalence class of DAGs can be represented by the partially directed graph Fig.34.3. These 3 DAGs can be proven to be OE in the following 3 ways:

1. Write the generic probability distributions represented by the 3 DAGs, and show that they are equal, as we did in Eq.34.2. That is the low brow way of proving OE.

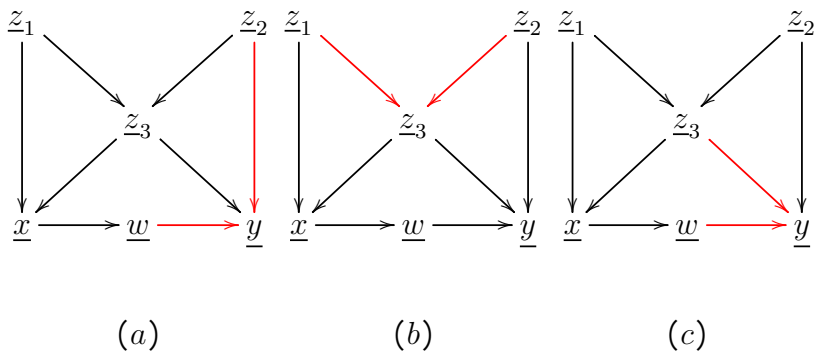


Figure 34.1: Example showing in red all v-structures of a particular DAG.

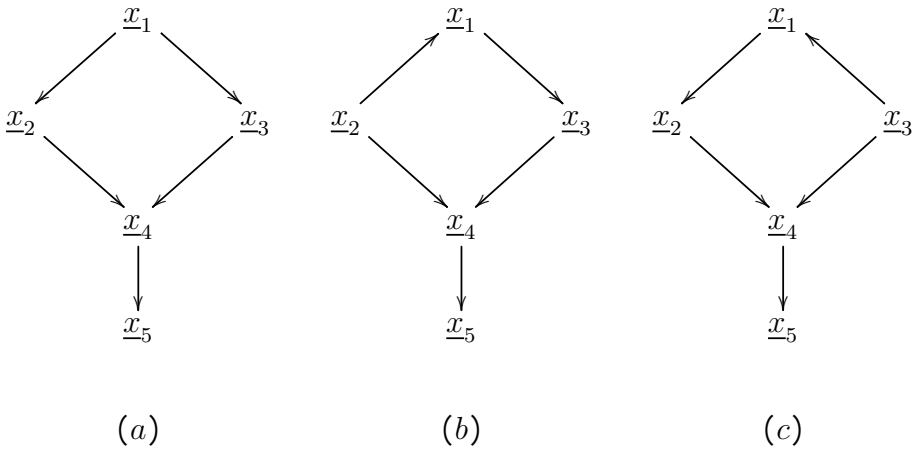


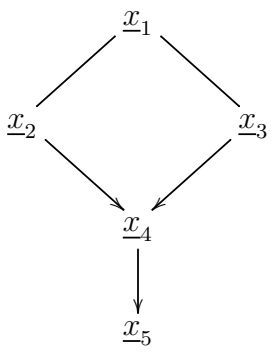
Figure 34.2: These 3 DAGs are observational equivalent (OE).

2. Use d-separation (see Chapter 12). Consider DAG (a) first. Rename the nodes as $\underline{\tau}_j$ with $j = 1, 2, \ldots$ so that the names are in topological order (i.e., so that the parents of $\underline{\tau}_j$ have indices that are smaller than j). The node names $\underline{x}_j$ of DAG (a) are already in topologigal order, so we skip this step for DAG (a). Now write down its total probability distribution and notice which parents of a fully connected DAG were omitted.

$$P(x_1, x_2, x_3, x_4, x_5) = \underbrace{P(x_5|x_4)}_{x_3, x_2, x_1 \text{ omitted}} \underbrace{P(x_4|x_3, x_2)}_{x_1 \text{ omitted}} \underbrace{P(x_3|x_1)}_{x_2 \text{ omitted}} P(x_2|x_1)P(x_1) \tag{34.3}$$

The observations of which parents were omitted can be stated in d-separation lingo as the following 3 orthogonality relations:[1]

[1] Normally, if we had changed from the original node names to the $\underline{\tau}_j$ node names, these orthogonality relations would first be stated in terms of the $\underline{\tau}_j$ names, and we could translate them so that they were stated in terms of the original node names. But for DAG (a) there was no need to use the $\underline{\tau}_j$ names.



(c)

Figure 34.3: This partially directed graph represents the 3 DAGs in Fig.34.2.

$$\underline{x}_3 \perp_P \underline{x}_2 \;\; | \;\; \underline{x}_1 \tag{34.4a}$$
$$\underline{x}_4 \perp_P \underline{x}_1 \;\; | \;\; \underline{x}_2, \underline{x}_3 \tag{34.4b}$$
$$\underline{x}_5 \perp_P (\underline{x}_1, \underline{x}_2, \underline{x}_3) \;\; | \;\; \underline{x}_4 \; . \tag{34.4c}$$

Going through the same procedure for the other 2 DAGs yields, for each of them, an equivalent set of 3 orthogonality equations.[2]

This is enough to conclude that the 3 DAGs of Fig.34.2 are OE.

Note that Eqs.(34.4) encompass all that there is to say about the observability of DAG (a). These 3 equations can be checked empirically to assess how well the DAG fits the data. For example, one can do OLS (ordinary least squares) regression $x_5 \sim x_1 + x_2 + x_3 + x_4$ on the data, i.e., try to fit $x_5 = \beta_0 + \sum_{i=1}^{4} \beta_i x_i$ to the data, and find that, to a good approximation, $\beta_1 = \beta_2 = \beta_3 = 0$.

3. Use the OE Theorem. All three DAGs have the same skeleton, and the same single v-structure $\underline{x}_2 \to \underline{x}_4 \leftarrow \underline{x}_3$.

[2] The $\underline{x}_j$ node names are no longer in topological order for DAGs (b) and (c) so for them you should go through the intermediate step of renaming the nodes $\underline{\tau}_j$, and then, after obtaining the orthogonality relations in terms of the $\underline{\tau}_j$ names, translating them back to the original $\underline{x}_j$ names.

Chapter 35

Program evaluation and review technique (PERT)

This chapter is based on Refs.[28] and [59].

PERT diagrams are used for scheduling a project consisting of a series of interdependent activities and estimating how long it will take to finish the project. PERT diagrams were invented by the NAVY in 1958 to manage a submarine project. Nowadays they are taught in many business and management courses.

A **PERT diagram** is a Directed Acyclic Graph (DAG) with the following properties. (See Fig.35.2 for an example of a PERT diagram). The nodes $\underline{E}_i$ for $i = 1, 2, \ldots, ne$ of a PERT diagram are called **events**. The edges $i \rightarrow j$ of a PERT diagram are called **activities**. An event represents the starting (kickoff) date of one or more activities. A PERT diagram has a single root node ($i = 1$, start event) and a single leaf node ($i = ne$, end event).

The PERT diagram user must initially provide a **Duration Times (DT) table** which gives $(DO_{i\rightarrow j}, DP_{i\rightarrow j}, DM_{i\rightarrow j})$ for each activity $i \rightarrow j$, where

$DO_{i\rightarrow j}$= optimistic duration time of activity $i \rightarrow j$

$DP_{i\rightarrow j}$= pessimistic duration time of activity $i \rightarrow j$

$DM_{i\rightarrow j}$= median duration time of activity $i \rightarrow j$

From the DT table, one calculates:

Duration time of activity $i \rightarrow j$

$$D_{i\rightarrow j} = \frac{1}{6}(DO_{i\rightarrow j} + DP_{i\rightarrow j} + 4DM_{i\rightarrow j}) \tag{35.1}$$

Duration Variance of activity $i \rightarrow j$

$$V_{i\rightarrow j} = \left(\frac{DO_{i\rightarrow j} - DP_{i\rightarrow j}}{DM_{i\rightarrow j}}\right)^2 \tag{35.2}$$

Often, it is convenient to define "dummy" edges with $D_{i\rightarrow j} = 0$. That is perfectly fine.

Define:

TES_i = Earliest start time for event i

TLS_i = Latest start time for event i

$slack_i = TLS_i - TES_i$ = slack for event i

$TEF_{i\to j} = TES_i + D_{i\to j}$ = Earliest finish time for activity $i \to j$.

$TLF_{i\to j} = TLS_j - D_{i\to j}$ = Latest finish time for activity $i \to j$. See footnote below. [1]

A **critical path** is a directed path (i.e., a chain of connected arrows, all pointing in the same direction) going from the start to the end node, such that slack equals zero at every node visited. In a DAG, the neighbors of a node is the union of its parent and children nodes. A critical path must also have all other nodes as neighbors; i.e, the union of the neighbors of every node in the path plus the nodes in the path itself, equals all nodes in the graph.

GOAL of PERT analysis: The main goal of PERT analysis is to find, based on the data of the DT table, the interval $[TES_i, TLS_i]$ giving a lower and an upper bound to the starting time of each node i. Another goal is to find a critical path for the PERT diagram (which represents an entire project). By adding the $D_{i\to j}$ of each edge of the critical path, one can get the mean value of the total duration of the entire project, and by adding the variances of each edge along the critical path, one can get an estimate of the total variance of the total duration. Knowing the mean and variance of the total duration and assuming a normal distribution, one can predict the probability that the actual duration will deviate by a certain amount from its mean.

To calculate the interval $[TES_i, TLS_i]$, one follows the following two steps.

1. Assume $TES_1 = 0$ and solve

$$TES_i = \max_{a\in pa(i)} (\underbrace{TES_a + D_{a\to i}}_{TEF_{a\to i}}) \tag{35.3}$$

for $i \in [2, ne]$. This recursive equation is solved by what is called "forward propagation", wherein one moves up the list of nodes i in order of increasing i starting at $i = 1$ with $TES_1 = 0$.

2. Assume $TLS_{ne} = TES_{ne}$ and solve

$$TLS_i = \min_{b\in ch(i)} (\underbrace{TLS_b - D_{i\to b}}_{TLF_{i\to b}}) \tag{35.4}$$

for $i \in [1, ne-1]$. This recursive equation is solved by what is called "backward propagation", wherein one moves down the list of nodes i in order of decreasing i starting at $i = ne$ with $TLS_{ne} = TES_{ne}$. TES_{ne} is known from step 1.

Eqs.(35.3) and (35.4) are illustrated in Fig.35.1.

[1] In the popular educational literature, the edge variables $TEF_{i\to j}$ and $TLF_{i\to j}$ are sometimes associated with the nodes, but they are clearly edge variables. This makes things confusing. The reason this is done is that some software draws PERT diagrams as trees whereas other software draws them as DAGs. For trees, storing $TEF_{i\to j}$ and $TLF_{i\to j}$ in a node makes some sense but not for DAGs. You will notice that giving specific names to the variables $TEF_{i\to j}$ and $TLF_{i\to j}$ is unnecessary. It is possible to delete all mention of their names from this chapter without losing any details. I only declare their names in this chapter so as tell the reader what they are in case he/she hears them mentioned and wonders what they are equal to in our notation.

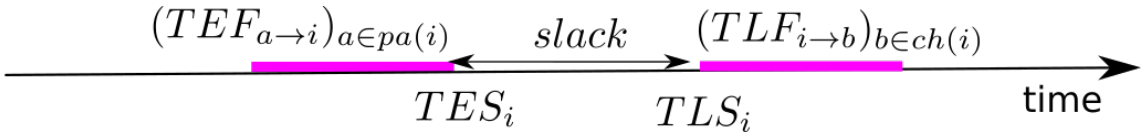


Figure 35.1: TES_i defined from info received from parents of i and TLS_i defined from info received from children of i.

Example

To illustrate PERT analysis, we end with an example. We present the example in the form of an exercise question and then provide the answer. This example comes from Ref.[28], except for part (e) about bnets, which is our own.

Question: For the PERT diagram of Fig.35.2, calculate the following:

(a) Interval $[TES_i, TLS_i]$ for all i.

(b) A critical path for this PERT diagram.

(c) The mean and variance of the total duration of the critical path.

(d) The probability that the total duration will be 225 days or less.

(e) A bnet interpretation of this problem.

Answer to (a) $[TES_i, TLS_i]$ are given by Fig.35.3.

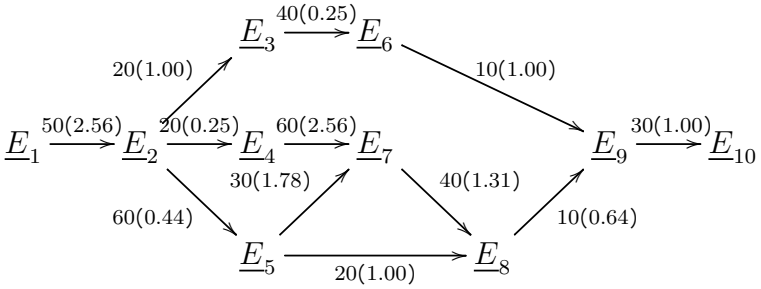
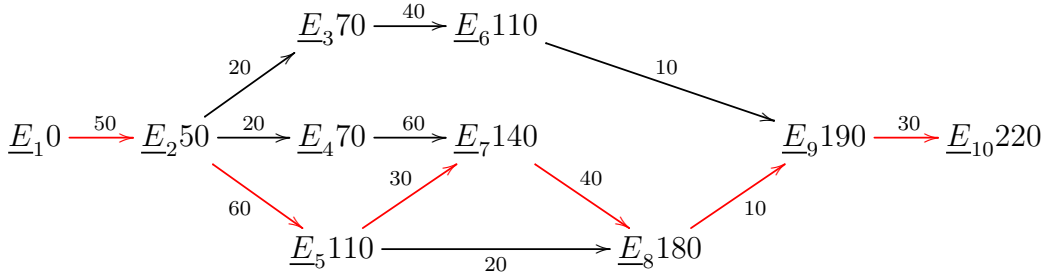
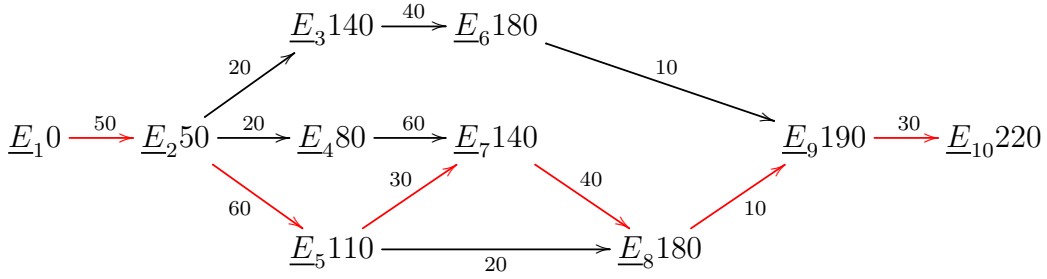


Figure 35.2: Example of a PERT diagram. The numbers attached to the arrows are the duration times $D_{i\to j}$ in days followed by, enclosed in parentheses, the variance $V_{i\to j}$ of that duration. The info given in this PERT diagram was derived from a DT table in Ref.[28]. The info in this PERT diagram is sufficient for calculating TES_i and TLS_i for each node i. The results of that calculation are given in Fig.35.3.

Answer to (b) The critical path is given in red in Fig.35.3. Note that this path does indeed have zero slack at each node it visits and the union of its neighborhood and the path itself encompasses all nodes.



TES_i (given after the node name) for node $\underline{E}_i$ for all i



TLS_i (given after the node name) for node $\underline{E}_i$ for all i

Figure 35.3: Results of calculating TES_i for all i via a forward pass, followed by calculating TLS_i for all i via a backward pass. Critical path indicated in red.

Answer to (c) The mean and variance of the total duration are calculated in Table 35.1.

Answer to (d)

$$P(\underline{x} < 225) = P\left[\frac{\underline{x} - \mu}{\sigma} \leq \frac{225 - 220}{\sqrt{7.73}}\right] \tag{35.5}$$

$$= P[\underline{z} \leq 1.80] \tag{35.6}$$

$$= 0.9641 \tag{35.7}$$

Answer to (e) Define 2 bnets.

1. The first PERT bnet is for calculating TES_i for all i and is given by Fig.35.4.
   The node TPMs, printed in blue, for the bnet Fig.35.4 are given by (this equation is to be evaluated recursively by a forward pass through the bnet):

$$P(TES_i | (TES_a)_{a \in pa(i)}) = \delta(TES_i, \max_{a \in pa(i)} (TES_a + D_{a \to i})) \tag{35.8}$$

2. The second PERT bnet is for calculating TLS_i for all i and is given by Fig.35.5. Note that the directions of all the arrows in the PERT diagram Fig.35.2 have been reversed so Fig.35.5 is a time reversed graph.

edge $i \to j$	duration $D_{i\to j}$	variance $V_{i\to j}$
A $(1 \to 2)$	50	2.56
D $(2 \to 5)$	60	0.44
G $(5 \to 7)$	30	1.78
J $(7 \to 8)$	40	1.31
K $(8 \to 9)$	10	0.64
L $(9 \to 10)$	30	1.00
Total	220	7.73

Table 35.1: Calculation of mean and variance of total duration along critical path.

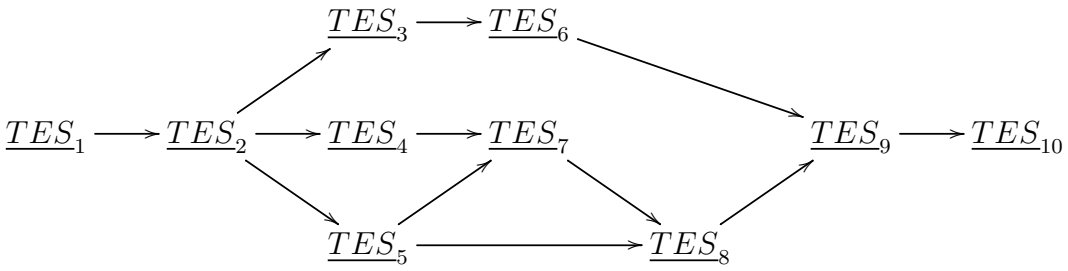


Figure 35.4: bnet for TES_i calculation.

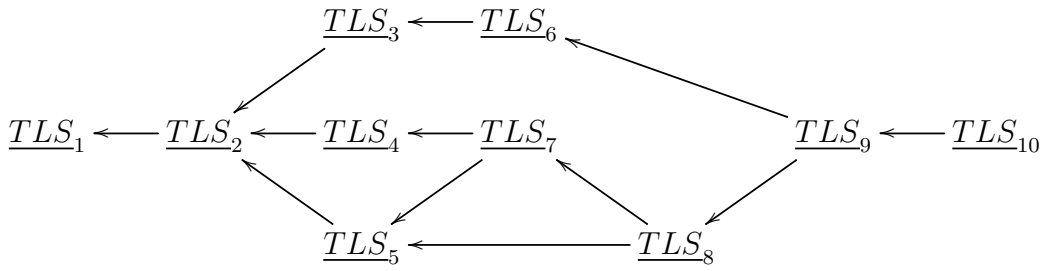


Figure 35.5: bnet for TLS_i calculation.

The node TPMs, printed in blue, for the bnet Fig.35.5 are given by (this equation is to be evaluate recursively by a backward pass through the bnet):

$$P(TLS_i|(TLS_b)_{b\in pa(i)}) = \delta(TLS_i, \min_{b\in pa(i)}(TLS_b - D^T_{b\to i})) \,, \tag{35.9}$$

where $D^T_{i\to j} = D_{j\to i}$.

Chapter 36

Recurrent Neural Networks

This chapter is mostly based on Ref.[14].

This chapter assumes you are familiar with the material and notation of Chapter 31 on plain Neural Nets.

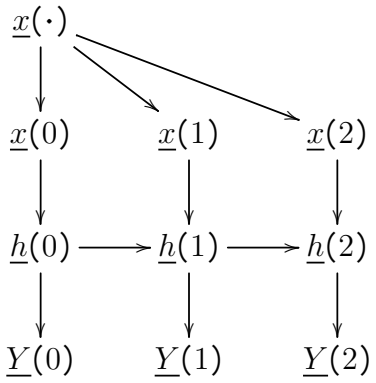


Figure 36.1: Simple example of RNN witb $T = 3$

Suppose
T is a positive integer.
$t = 0, 1, \ldots, T-1$,
$\underline{x}_i(t) \in \mathbb{R}$ for $i = 0, 1, \ldots, numx - 1$,
$\underline{h}_i(t) \in \mathbb{R}$ for $i = 0, 1, \ldots, numh - 1$,
$\underline{Y}_i(t) \in \mathbb{R}$ for $i = 0, 1, \ldots, numy - 1$,
$W^{h|x} \in \mathbb{R}^{numh \times numx}$,
$W^{h|h} \in \mathbb{R}^{numh \times numh}$,
$W^{y|h} \in \mathbb{R}^{numy \times numh}$,
$b^y \in \mathbb{R}^{numy}$,
$b^h \in \mathbb{R}^{numh}$.
Henceforth, $x(\cdot)$ will mean the array of $x(t)$ for all t.

The simplest kind of recurrent neural network (RNN) has the bnet Fig.36.1 with arbitrary T. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(x(\cdot)) = \text{given} \tag{36.1}$$

$$P(x(t)) = \delta(x(t), [x(\cdot)]_t) \tag{36.2}$$

$$P(h(t) \mid h(t-1), x(t)) = \delta(h(t), \mathcal{A}(W^{h|x} x(t) + W^{h|h} h(t-1) + b^h)) \ , \tag{36.3}$$

where $h(-1) = 0$.

$$P(Y(t) \mid h(t)) = \delta(Y(t), \mathcal{A}(W^{y|h} h(t) + b^y)) \tag{36.4}$$

Define

$$W^h = [W^{h|x}, W^{h|h}, b^h] \ , \tag{36.5}$$

and

$$W^y = [W^{y|h}, b^y] \ . \tag{36.6}$$

The bnet of Fig.36.1 can be used for classification once its parameters W^h and W^y have been optimized. To optimize those parameters via gradient descent, one can use the bnet of Fig.36.2.

Let $\sigma = 0, 1, \ldots, nsam(\vec{x}) - 1$ be the labels for a minibatch of samples. The node TPMs, printed in blue, for bnet Fig.36.2, are as follows.

$$P(x(\cdot)[\sigma]) = \text{given} \tag{36.7}$$

$$P(x(t)[\sigma]) = \delta(x(t)[\sigma], [x(\cdot)]_t[\sigma]) \tag{36.8}$$

$$P(h(t)[\sigma] \mid h(t-1)[\sigma], x(t)[\sigma]) = \delta(h(t)[\sigma], \mathcal{A}(W^{h|x} x(t)[\sigma] + W^{h|h} h(t-1)[\sigma] + b^h) \tag{36.9}$$

$$P(Y(t)[\sigma] \mid h(t-1)[\sigma]) = \delta(Y(t)[\sigma], \mathcal{A}(W^{y|h} h(t-1)[\sigma] + b^y) \tag{36.10}$$

$$P(y(\cdot)[\sigma] \mid x(\cdot)[\sigma]) = \text{given} \tag{36.11}$$

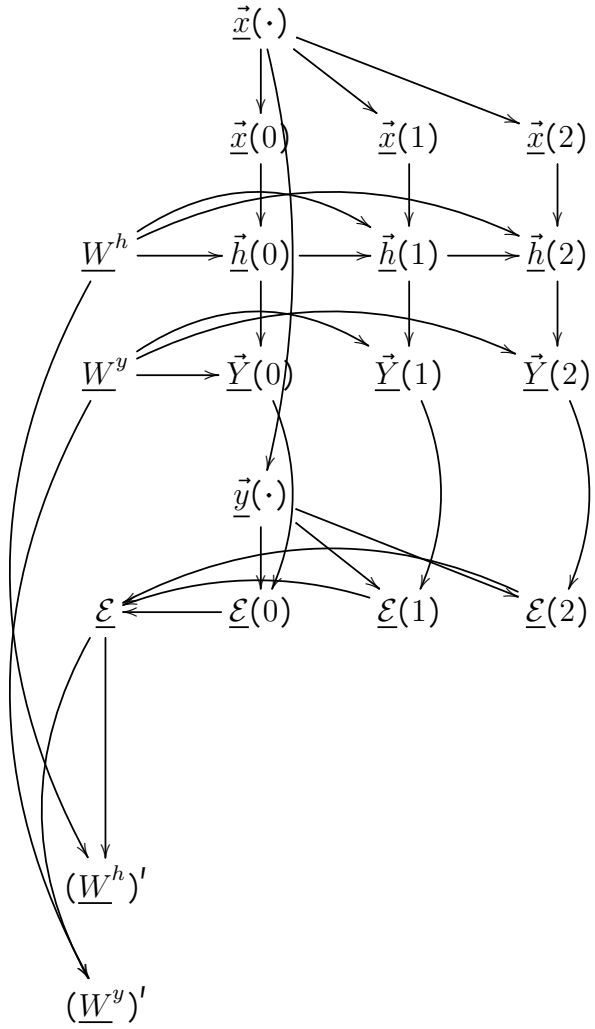


Figure 36.2: RNN bnet used to optimize parameters W^h and W^y of RNN bnet Fig.36.1.

$$P(\mathcal{E}(t) \mid \vec{y}(\cdot), \vec{Y}(t)) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} d(y(t)[\sigma], Y(t)[\sigma]) \ , \tag{36.12}$$

where

$$d(y, Y) = |y - Y|^2 \ . \tag{36.13}$$

If $y, Y \in [0, 1]$, one can use this instead

$$d(y, Y) = XE(y \to Y) = -y \ln Y - (1 - y) \ln(1 - Y) \ . \tag{36.14}$$

$$P(\mathcal{E} \mid [\mathcal{E}(t)]_{\forall t}) = \delta(\mathcal{E}, \sum_t \mathcal{E}(t)) \tag{36.15}$$

For $a = h, y$,

$$P(W^a) = \text{given} . \tag{36.16}$$

The first time it is used, W^a is fairly arbitrary. Afterwards, it is determined by previous horizontal stage.

$$P((W^a)'|\mathcal{E}, W^a) = \delta((W^a)', W^a - \eta^a \partial_{W^a}\mathcal{E}) . \tag{36.17}$$

$\eta^a > 0$ is the learning rate for W^a.

Language Sequence Modeling

Figs.36.1, and 36.2 with arbirary T can be used as follows to do Language Sequence Modeling.

For this usecase, one must train with the following TPM for node $\vec{y}(\cdot)$:

$$P(y(\cdot)[\sigma] \mid x(\cdot)[\sigma]) = \prod_t \mathbb{1}(\quad y(t)[\sigma] = P(x(t)[\sigma] \mid [x(t')[\sigma]]_{t'<t}) \quad) \tag{36.18}$$

With such training, one gets

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = P(x(t) \mid [x(t')]_{t'<t}) \quad) . \tag{36.19}$$

Therefore,

$$Y(0) = P(x(0)) , \tag{36.20}$$

$$Y(1) = P(x(1)|x(0)) , \tag{36.21}$$

$$Y(2) = P(x(2)|x(0), x(1)) , \tag{36.22}$$

and so on.

We can use this to:

- predict the probability of a sentence,
  example: Get $P(x(0), x(1), x(2))$.
- predict the most likely next word in a sentence,
  example: Get $P(x(2)|x(0), x(1))$.

- generate fake sentences.

  example:

  Get $x(0) \sim P(x(0))$.

  Next get $x(1) \sim P(x(1)|x(0))$.

  Next get $x(2) \sim P(x(2)|x(0), x(1))$.

Other types of RNN

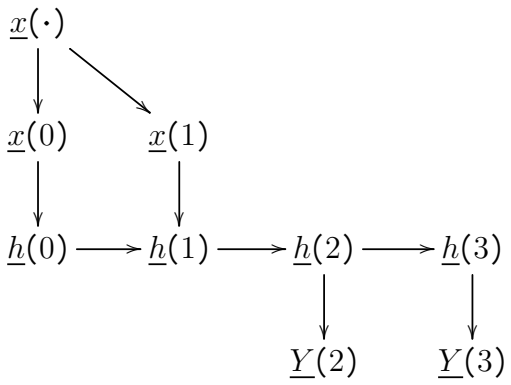


Figure 36.3: RNN bnet of the many to many kind. This one can be used for translation. $x(0)$ and $x(1)$ might denote two words of an English sentence, and $Y(2)$ and $Y(3)$ might be their Italian translation.

Let $\mathcal{T} = \{0, 1, \ldots, T-1\}$, and $\mathcal{T}^x, \mathcal{T}^y \subset \mathcal{T}$. Above, we assumed that $\underline{x}(t)$ and $\underline{Y}(t)$ were both defined for all $t \in \mathcal{T}$. More generally, they might be defined only for subsets of $\mathcal{T}$: $\underline{x}(t)$ for $t \in \mathcal{T}^x$ and $\underline{Y}(t)$ for $t \in \mathcal{T}^y$. If $|\mathcal{T}^x| = 1$ and $|\mathcal{T}^y| > 1$, we say the RNN bnet is of the 1 to many kind. In general, can have **1 to 1, 1 to many, many to 1, many to many** RNN bnets.

Plain RNNs can suffer from the **vanishing or exploding gradients problem**. There are various ways to mitigate this (good choice of initial W^h and W^y, good choice of activation functions, regularization). Or by using GRU or LSTM (discussed below). **GRU and LSTM** were designed to mitigate the vanishing or exploding gradients problem. They are very popular in NLP (Natural Language Processing).

Long Short Term Memory (LSTM) unit (1997)

This section is based on Wikipedia article Ref.[50]. In this section, $\odot$ will denote the Hadamard matrix product (elementwise product).

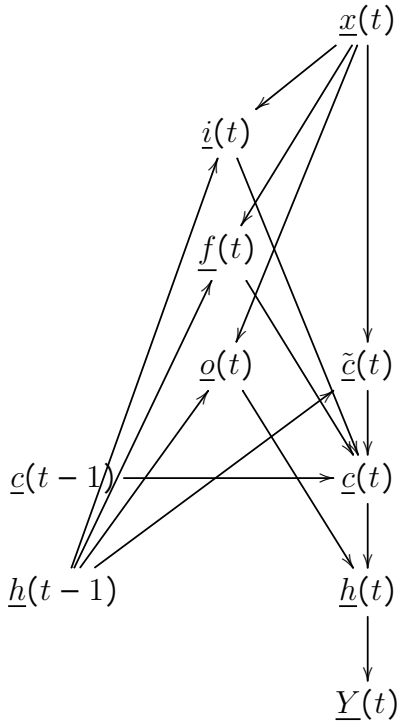


Figure 36.4: bnet for a Long Short Term Memory (LSTM) unit.

Let
$\underline{x}(t) \in \mathbb{R}^{numx}$: input vector to the LSTM unit
$\underline{f}(t) \in \mathbb{R}^{numh}$: forget gate's activation vector
$\underline{i}(t) \in \mathbb{R}^{numh}$: input/update gate's activation vector
$\underline{o}(t) \in \mathbb{R}^{numh}$: output gate's activation vector
$\underline{h}(t) \in \mathbb{R}^{numh}$: hidden state vector also known as output vector of the LSTM unit
$\underline{\tilde{c}}(t) \in \mathbb{R}^{numh}$: cell input activation vector
$\underline{c}(t) \in \mathbb{R}^{numh}$: cell state vector
$\underline{Y}(t) \in \mathbb{R}^{numy}$: classification of $x(t)$.
$W \in \mathbb{R}^{numh \times numx}$, $U \in \mathbb{R}^{numh \times numh}$ and $b \in \mathbb{R}^{numh}$: weight matrices and bias vectors, parameters learned by training.
$\mathcal{W}^{y|h} \in \mathbb{R}^{numy \times numh}$: weight matrix

Fig.36.4 is a bnet net for a LSTM unit. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(f(t)|x(t),h(t-1)) = \mathbb{1}(\quad f(t) = \text{sig}(W^{f|x}x(t) + U^{f|h}h(t-1) + b^f)\quad),\tag{36.23}$$

where $h(-1) = 0$.

$$P(i(t)|x(t),h(t-1)) = \mathbb{1}(\quad i(t) = \text{sig}(W^{i|x}x(t) + U^{i|h}h(t-1) + b^i)\quad)\tag{36.24}$$

$$P(o(t)|x(t),h(t-1)) = \mathbb{1}(\quad o(t) = \text{sig}(W^{o|x}x(t) + U^{o|h}h(t-1) + b^o)\quad)\tag{36.25}$$

$$P(\tilde{c}(t)|x(t),h(t-1)) = \mathbb{1}(\quad \tilde{c}(t) = \tanh(W^{c|x}x(t) + U^{c|h}h(t-1) + b^c)\quad)\tag{36.26}$$

$$P(c(t)|f(t),c(t-1),i(t),\tilde{c}(t)) = \mathbb{1}(\quad c(t) = f(t) \odot c(t-1) + i(t) \odot \tilde{c}(t)\quad)\tag{36.27}$$

$$P(h(t)|o(t),c(t)) = \mathbb{1}(\quad h(t) = o(t) \odot \tanh(c(t))\quad)\tag{36.28}$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h}h(t) + b^y)\quad)\tag{36.29}$$

Gated Recurrence Unit (GRU) (2014)

This section is based on Wikipedia article Ref.[42]. In this section, $\odot$ will denote the Hadamard matrix product (elementwise product).

GRU is a more recent (17 years later) attempt at simplifying LSTM unit.

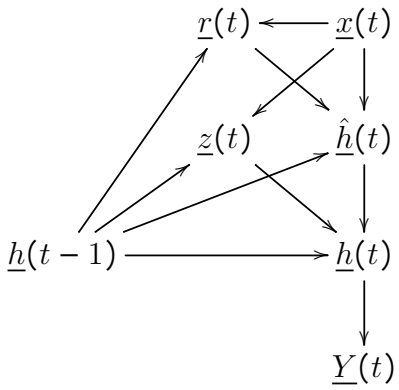


Figure 36.5: bnet for a Gated Recurrent Unit (GRU).

Let
$\underline{x}(t) \in \mathbb{R}^{numx}$: input vector
$\underline{h}(t) \in \mathbb{R}^{numh}$: output vector
$\underline{\hat{h}}(t) \in \mathbb{R}^{numh}$: candidate activation vector
$\underline{z}(t) \in \mathbb{R}^{numh}$: update gate vector
$\underline{r}(t) \in \mathbb{R}^{numh}$: reset gate vector
$\underline{Y}(t) \in \mathbb{R}^{numy}$: classification of $x(t)$.
$W \in \mathbb{R}^{numh \times numx}$, $U \in \mathbb{R}^{numh \times numh}$ and $b \in \mathbb{R}^{numh}$: weight matrices and bias vectors, parameters learned by training.
$\mathcal{W}^{y|h} \in \mathbb{R}^{numy \times numh}$: weight matrix

Fig.36.5 is a bnet net for a GRU. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(z(t)|x(t), h(t-1)) = \mathbb{1}(\quad z(t) = \text{sig}(W^{z|x}x(t) + U^{z|h}h(t-1) + b^z) \quad), \tag{36.30}$$

where $h(-1) = 0$.

$$P(r(t)|x(t), h(t-1)) = \mathbb{1}(\quad r(t) = \text{sig}(W^{r|x}x(t) + U^{r|h}h(t-1) + b^r) \quad) \tag{36.31}$$

$$P(\hat{h}(t)|x(t), r(t), h(t-1)) = \mathbb{1}(\quad \hat{h}(t) = \tanh(W^{h|x}x(t) + U^{h|h}(r(t) \odot h(t-1)) + b^h) \tag{36.32}$$

$$P(h(t)|z(t), h(t-1), \hat{h}(t)) = \mathbb{1}(\quad h(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot \hat{h}(t) \quad) \tag{36.33}$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h} h(t) + b^{y}) \quad) \tag{36.34}$$

Chapter 37

Reinforcement Learning (RL)

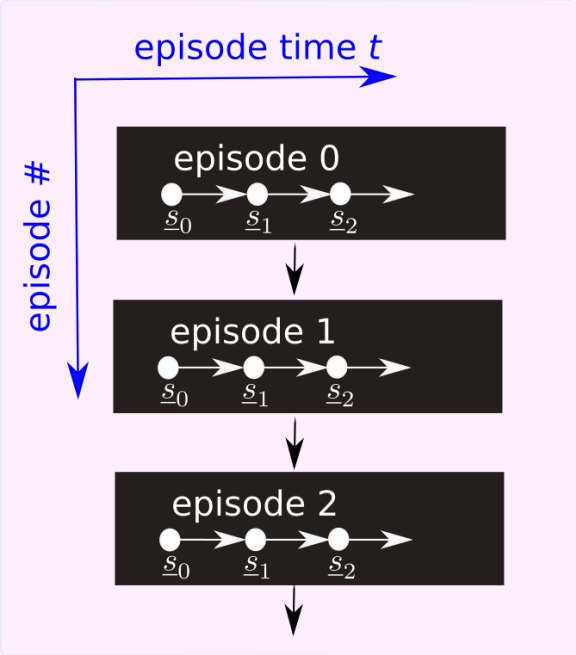


Figure 37.1: Axes for episode time and episode number.

I based this chapter on the following references. Refs.[3][9]

In RL, we consider an "agent" or robot that is learning.

Let $T \in \mathbb{Z}_{>0}$ be the duration time of an **episode** of learning. If $T = \infty$, we say that the episode has an infinite time horizon. A learning episode will evolve towards the right, for times $t = 0, 1, \ldots, T-1$. We will consider multiple learning episodes. The episode number will evolve from top to bottom. This is illustrated in Fig.37.1.

Let $\underline{s}_t \in S_{\underline{s}}$ for $t \in [0, T-1]_{\mathbb{Z}}$ be random variables that record the **state** of the agent at various times t.

Let $\underline{a}_t \in S_{\underline{a}}$ for $t \in [0, T-1]_{\mathbb{Z}}$ be random variables that record the **action** of the agent at various times t.

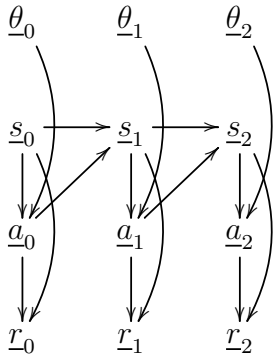


Figure 37.2: State-Action-Reward dynamical bnet

Let $\underline{\theta}_t \in S_{\underline{\theta}}$ for $t \in [0, T-1]_{\mathbb{Z}}$ be random variables that record the **policy parameters** at various times t.

For $\underline{X} \in \{\underline{s}, \underline{a}, \underline{\theta}\}$, define $\underline{X}$ followed by a dot to be the vector

$$\underline{X}_\cdot = [\underline{X}_0, \underline{X}_1, \ldots, \underline{X}_{T-1}] \,. \tag{37.1}$$

Also let

$$\underline{X}_{\geq t} = [\underline{X}_t, \underline{X}_{t+1}, \ldots, \underline{X}_{T-1}] \,. \tag{37.2}$$

Fig.37.2 shows the basic State-Action-Reward bnet for an agent that is learning. The TPMs for the nodes of Fig.37.2 are given in blue below:

$$P(a_t|s_t, \theta_t) = \text{given.} \tag{37.3}$$

$P(a_t|s_t, \theta_t)$ is called a **policy with parameter θ_t.**

$$P(s_t|s_{t-1}, a_{t-1}) = \text{given.} \tag{37.4}$$

$P(s_t|s_{t-1}, a_{t-1})$ is called the **TPM of the model**. $P(s_t|s_{t-1}, a_{t-1})$ reduces to $P(s_0)$ when $t = 0$.

$$P(r_t|s_t, a_t) = \delta(r_t, r(s_t, a_t))) \,. \tag{37.5}$$

$r : S_{\underline{s}} \times S_{\underline{a}} \to \mathbb{R}$ is a given **one-time reward function**.

Note that

$$P(s_\cdot, a_\cdot|\theta_\cdot) = \prod_{t=0}^{T-1} \{P(s_t|s_{t-1}, a_{t-1}) P(a_t|s_t, \theta_t)\} \,. \tag{37.6}$$

Define the **all times reward** Σ by

$$\Sigma(s_\cdot, a_\cdot) = \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \,. \tag{37.7}$$

Here $0 < \gamma < 1$. γ, called the **discount rate**, is included to assure convergence of Σ when $T \to \infty$. If $r(s_t, a_t) < K$ for all t, then $\Sigma < K\frac{1}{1-\gamma}$.

Define the **objective (i.e. goal) function** $E\Sigma(\theta_\cdot)$ by

$$E\Sigma(\theta_\cdot) = E_{\underline{s}_\cdot,\underline{a}_\cdot|\theta_\cdot}\Sigma(\underline{s}_\cdot, \underline{a}_\cdot) = \sum_{s_\cdot,a_\cdot} P(s_\cdot, a_\cdot|\theta_\cdot)\Sigma(s_\cdot, a_\cdot) \tag{37.8}$$

The goal of RL is to maximize the objective function over its parameters $\theta_\cdot$. The parameters $\theta_\cdot^*$. that maximize the objective function are the optimum strategy:

$$\theta_\cdot^* = \underset{\theta_\cdot}{\operatorname{argmax}}\, E\Sigma(\theta_\cdot) \tag{37.9}$$

Define a **future reward** for times $\geq t$ as:

$$\Sigma_{\geq t}((s_{t'}, a_{t'})_{t'\geq t}) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) \tag{37.10}$$

Define the following **expected conditional future rewards** (rewards for times $\geq t$, conditioned on certain quantities having given values):

$$v_t = v(s_t, a_t; \theta_\cdot) = E_{\underline{s}_\cdot,\underline{a}_\cdot|s_t,a_t,\theta_\cdot}[\Sigma_{\geq t}] \tag{37.11}$$

$$V_t = V(s_t; \theta_\cdot) = E_{\underline{s}_\cdot,\underline{a}_\cdot|s_t,\theta_\cdot}[\Sigma_{\geq t}] = E_{\underline{a}_t|s_t,\theta_\cdot}[v(s_t, \underline{a}_t; \theta_\cdot)] \tag{37.12}$$

v is usually called Q in the literature. We will refer to Q as v in order to follow a convention wherein an $\underline{a}_t$-average changes a lower case letter to an upper case one.

We will sometimes write $v(s_t, a_t)$ instead of $v(s_t, a_t; \theta_\cdot)$.

Since $E\Sigma_{\geq t}$ only depends on $\theta_{\geq t}$, $v(s_t, a_t; \theta_\cdot) = v(s_t, a_t; \theta_{\geq t})$, and $V(s_t; \theta_\cdot) = V(s_t; \theta_{\geq t})$.

Note that the objective function $E\Sigma$ can be expressed in terms of v_0 by averaging over its unaveraged parameters:

$$E\Sigma(\theta_\cdot) = E_{\underline{s}_0,\underline{a}_0|\theta_0} v(\underline{s}_0, \underline{a}_0; \theta_\cdot) \tag{37.13}$$

Define a **one-time reward** and an **expected conditional one-time reward** as:

$$r_t = r(s_t, a_t) \tag{37.14}$$

$$R_t = R(s_t; \theta_t) = E_{\underline{a}_t|s_t,\theta_t}[r(s_t, \underline{a}_t)] \ . \tag{37.15}$$

Note that

$$\Sigma_{\geq t} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-1-t} r_{t+(T-1-t)} \tag{37.16}$$

$$= r_t + \gamma\Sigma_{\geq t+1}; . \tag{37.17}$$

If we take $E_{\underline{s}_\cdot,\underline{a}_\cdot|s_t,a_t,\theta_\cdot}[\cdot]$ of both sides of Eq.(37.17), we get

$$v_t = r_t + \gamma E_{\underline{s}_{t+1},\underline{a}_{t+1}|\theta_\cdot}[v_{t+1}] \ . \tag{37.18}$$

If we take $E_{\underline{s}_\cdot,\underline{a}_\cdot|s_t,\theta_\cdot}[\cdot]$ of both sides of Eq.(37.17), we get

$$V_t = R_t + \gamma E_{\underline{s}_{t+1}|\theta_\cdot}[V_{t+1}] \; . \tag{37.19}$$

Note that

$$\begin{aligned} \Delta r_t &= r_t - R_t && (37.20)\\ &= r_t - (V_t - \gamma E_{\underline{s}_{t+1}|\theta_\cdot}[V_{t+1}]) && (37.21)\\ &= r_t + \gamma E_{\underline{s}_{t+1}|\theta_\cdot}[V_{t+1}] - V_t \; . && (37.22) \end{aligned}$$

Define

$$\Delta v_t = v_t - V_t \; . \tag{37.23}$$

Note that

$$\Delta v_t = \Delta r_t \; . \tag{37.24}$$

Next, we will discuss 3 RL bnets

- exact RL bnet (exact, assumes policy is known)
- Actor-Critic RL bnet (approximate, assumes policy is known)
- Q function learning RL bnet (approximate, assumes policy is NOT known)

Exact RL bnet

An exact RL bnet is given by Fig.37.3.

Fig.37.3 is the same as Fig.37.2 but with more nodes added in order to optimize the policy parameters. Here are the TPMs, printed in blue, for the nodes not already discussed in connection to Fig.37.2.

$$P(\theta_t|\theta_\cdot) = \delta(\theta_t, (\theta_\cdot)_t) \tag{37.25}$$

$$\forall (s_t, a_t): \;\; P(v_t(s_t,a_t)|r_t, v_{t+1}(\cdot), \theta_\cdot) = \delta(v_t(s_t,a_t), r_t + \gamma E_{\underline{s}_{t+1},\underline{a}_{t+1}|\theta_\cdot}[v_{t+1}]) \tag{37.26}$$

$$P(\theta_\cdot'|\theta_\cdot, v_0(\cdot)) = \delta(\theta_\cdot', \theta_\cdot + \alpha \partial_{\theta_\cdot} \underbrace{E_{\underline{s}_0,\underline{a}_0|\theta_0} v(\underline{s}_0, \underline{a}_0; \theta_\cdot)}_{E\Sigma(\theta_\cdot)}) \tag{37.27}$$

$\alpha > 0$ is called the **learning rate**. This method of improving $\theta_\cdot$ is called gradient ascent.

Concerning the gradient of the objective function, note that

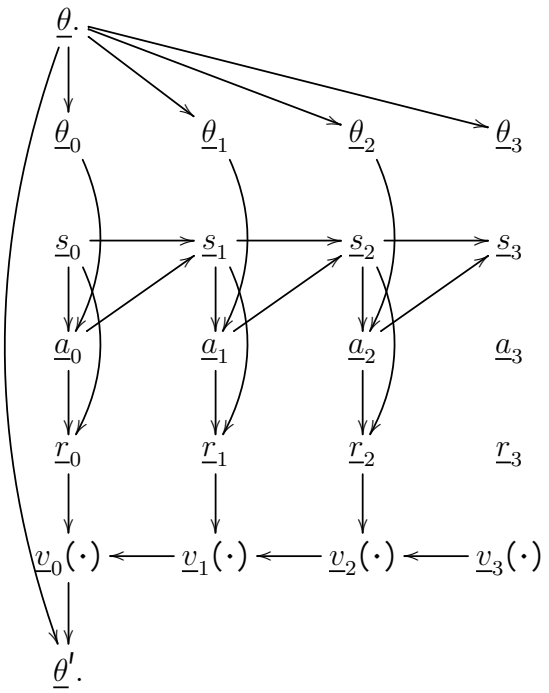


Figure 37.3: Exact RL bnet. $v_t(\cdot)$ means the array $[v_t(s_t, a_t)]_{\forall s_t, a_t}$ The following arrows are implicit: for all t, arrow from $\underline{\theta}. \rightarrow \underline{v}_t(\cdot)$. We did not draw those arrows so as not to clutter the diagram.

$$\partial_{\theta_t} E\Sigma(\theta.) = \sum_{s.,a.} \partial_{\theta_t} P(s.,a.|\theta.)\Sigma(s.,a.) \tag{37.28}$$

$$= \sum_{s.,a.} P(s.,a.|\theta.)\partial_{\theta_t} \ln P(s.,a.|\theta.)\Sigma(s.,a.) \tag{37.29}$$

$$= E_{\underline{s}.,\underline{a}.|\theta.}\left\{\partial_{\theta_t} \ln P(a_t|s_t,\theta_t)\Sigma(s.,a.)\right\} \ . \tag{37.30}$$

If we run the agent $nsam(\vec{s}_t)$ times and obtain samples $s_t[i], a_t[i]$ for all t and for $i = 0, 1, \ldots, nsam(\vec{s}_t) - 1$, we can express this gradient as follows:

$$\partial_{\theta_t} E\Sigma(\theta.) \approx \frac{1}{nsam(\vec{s}_t)} \sum_i \sum_{t=0}^{T-1} \partial_{\theta_t} \ln P(a_t[i] \mid s_t[i], \theta_t) r(s_t[i], a_t[i]) \ . \tag{37.31}$$

The exact RL bnet Fig.37.3 is difficult to use to calculate the optimum parameters θ^*.. The problem is that $\underline{s}_t$ propagates towards the future and the $\underline{v}_t(\cdot)$ propagates towards the past, so we don't have a Markov Chain with a chain link for each t (i.e., a dynamical bnet) in the episode time direction. Hence, people have come up with approximate RL bnets that are doubly dynamical (i.e., dynamical along the episode time and episode number axes.) We discuss some of those approximate RL bnets next.

Actor-Critic RL bnet

For the actor-critic RL bnet, we approximate Eq.(37.31) by

$$\partial_{\theta_t} E\Sigma(\theta.) \approx \frac{1}{nsam(\vec{s})} \sum_i \sum_{t=0}^{T-1} \underbrace{\partial_{\theta_t} \ln P(a_t[i] \mid s_t[i], \theta_t)}_{Actor} \underbrace{\Delta r_t(s_t[i], a_t[i])}_{Critic} \tag{37.32}$$

The actor-critic RL bnet is given by Fig.37.4. This bnet is approximate and assumes that the policy is known. The TPMs for its nodes are given in blue below.

$$P(\theta_t) = \text{given} \tag{37.33}$$

$$P(s_t[i] \mid s_{t-1}[i], a_{t-1}[i]) = \text{given} \tag{37.34}$$

$$P(a_t[i] \mid s_t[i], \theta_t) = \text{given} \tag{37.35}$$

$$P(r_t[i] \mid s_t[i], a_t[i]) = \delta(r_t[i], r(s_t[i], a_t[i])) \tag{37.36}$$

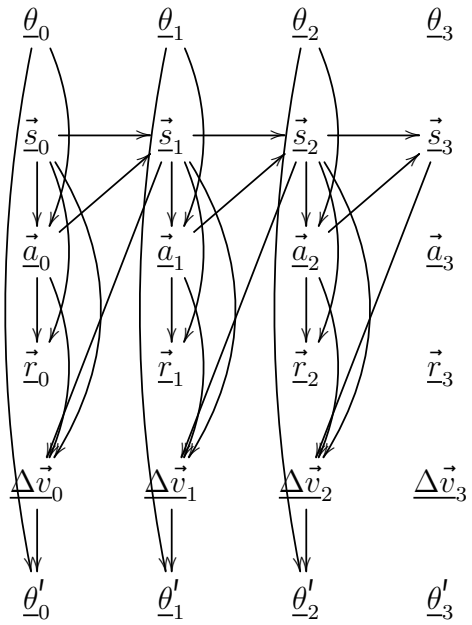


Figure 37.4: Actor-Critic RL bnet.

$r : S_{\underline{s}} \times S_{\underline{a}} \to \mathbb{R}$ is given.

$$P(\Delta v_t[i] \mid s_t[i], a_t[i], s_{t+1}[i]) = \delta(\Delta v_t[i], r(s_t[i], a_t[i]) + \gamma \hat{V}(s_{t+1}[i]; \phi') - \hat{V}(s_t[i]); \phi) \ . \quad (37.37)$$

$$P(\theta'.) = \delta(\theta'., \theta_t + \alpha \partial_{\theta_t} \sum_i \ln P(a_t[i] \mid s_t[i], \theta_t) \Delta v_t[i]) \quad (37.38)$$

$\hat{V}(s_t[i]); \phi)$ is obtained by curve fitting (see Chapter 4) using samples $(s_t[i], a_t[i])\ \forall t, i$ with

$$y[i] = \sum_{t'=t}^{T} r(s_{t'}[i], a_{t'}[i]) \quad (37.39)$$

and

$$\hat{y}[i] = \hat{V}(s_t[i]; \phi) \ . \quad (37.40)$$

Eq.(37.39) is an approximation because $(s_{t'}, a_{t'})_{t'>t}$ are averaged over in the exact expression for $V(s_t)$. $\hat{V}(s_{t+1}[i]); \phi')$ is obtained in the same way as $\hat{V}(s_t[i]); \phi)$ but with t replaced by $t+1$ and ϕ by ϕ'.

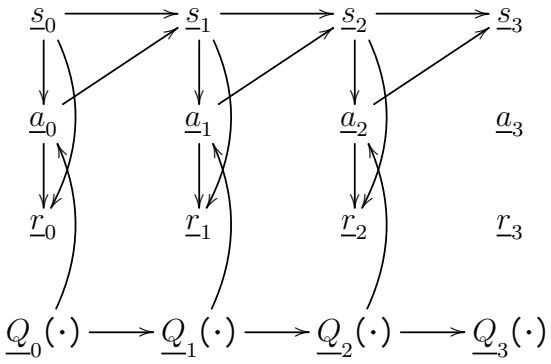


Figure 37.5: Q function learning RL bnet.

Q function learning RL bnet

The Q-function learning RL bnet is given by Fig.37.5. This bnet is approximate and assumes that the policy is NOT known. The TPMs for its nodes are given in blue below. (Remember that $Q = v$).

$$P(s_t|s_{t-1}, a_{t-1}) = \text{ given} \tag{37.41}$$

$$P(a_t|s_t, v_t(\cdot)) = \delta(a_t, \underset{a}{\text{argmax}}\, v_t(s_t, a)) \tag{37.42}$$

$$P(r_t|s_t, a_t) = \delta(r_t, r(s_t, a_t)) \tag{37.43}$$

$r : S_{\underline{s}} \times S_{\underline{a}} \to \mathbb{R}$ is given.

$$\begin{aligned} \forall (s_t, a_t): \quad & P(v_t(s_t, a_t)|v_{t-1}(\cdot)) = \\ & = \delta(v_t(s_t, a_t), r(s_t, a_t) + \gamma \text{max}_a E_{\underline{s}_{t+1}|s_t, a_t} v_{t-1}(\underline{s}_{t+1}, a)) \end{aligned} \tag{37.44}$$

This value for $v_t(s_t, a_t)$ approximates $v_t = r_t + \gamma E_{\underline{s}_{t+1}, \underline{a}_{t+1}} v_{t+1}$.

Some people use the bnet of Fig.37.6) instead of Fig.37.5 and replace Eq.(37.44) by

$$\begin{aligned} \forall (s_t, a_t): \quad & P(v_t(s_t, a_t)|s_{t+1}, v_{t-1}(\cdot)) = \\ & = \delta(v_t(s_t, a_t), r(s_t, a_t) + \gamma \text{max}_a v_{t-1}(s_{t+1}, a)) \,. \end{aligned} \tag{37.45}$$

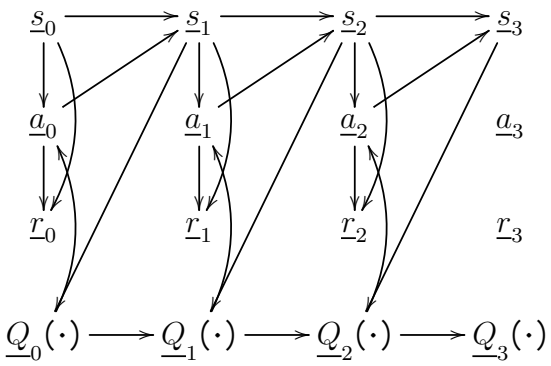


Figure 37.6: Q function learning RL bnet. Same as Fig.37.5 but with new arrow passing s_t to Q_{t-1}.

Chapter 38

Reliability Box Diagrams and Fault Tree Diagrams

This chapter is based on Refs.[23] and [31].

In this chapter, we assume that reader is familiar with Boolean Algebra. See the Notational Conventions Chapter 0.3 for a quick review of what we recommend that you know about Boolean Algebra to fully appreciate this chapter.

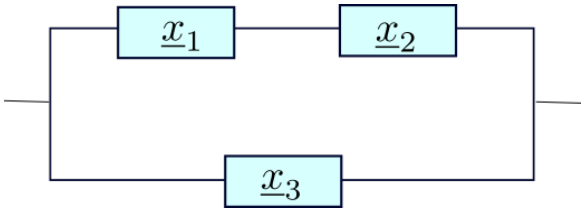


Figure 38.1: Example of rbox diagram.

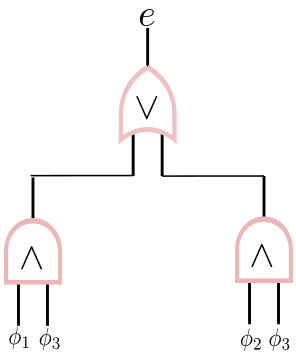


Figure 38.2: An ftree diagram equivalent to Fig.38.1. It represents $e = (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)$.

Complicated devices with a large number of components such as airplanes or rockets can fail in many ways. If their performance depends on some components working in series and one of the

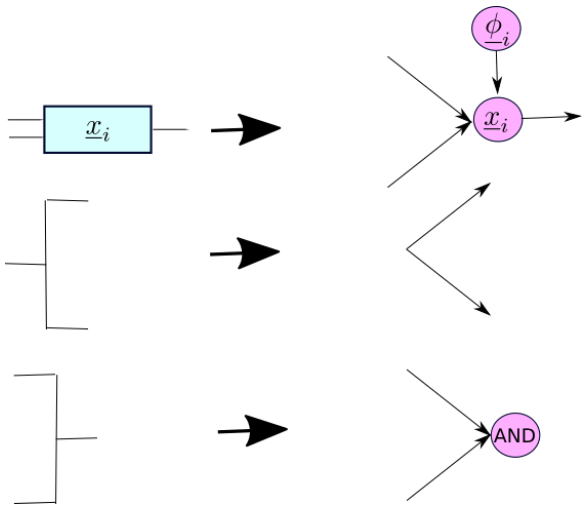


Figure 38.3: How to map an rbox diagram to a bnet.

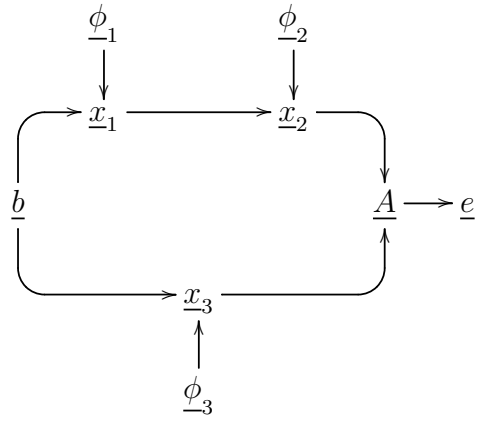


Figure 38.4: bnet corresponding the rbox diagram Fig.38.1.

components in the series fails, this may lead to catastrophic failure. To avert such disasters, engineers use equivalent components connected in parallel instead of in series, thus providing multiple backup systems. They analyze the device to find its weak points and add backup capabilities there. They also estimate the average time to failure for the device.

The two most popular diagrams for finding the failure modes and their rates for large complicated devices are

- rbox diagrams = Reliability Box diagrams. See Fig.38.1 for an example.
- ftree diagrams = Fault Tree Diagrams. See Fig.38.2 for an example.

In an ftree diagram, several nodes might stand for the same component of a physical device. In an rbox diagram, on the other hand, each node represents a distinct component in a device. Hence,

rbox diagrams resemble the device they are addressing whereas ftree diagrams don't. Henceforth, we will refer to this desirable property as **physical resemblance**.

As we will show below with an example, it is pretty straightforward to translate an rbox to an ftree diagram. Going the other way, translating an ftree to an rbox diagram is much more difficult.

Next we will define a new kind of bnet that we will call a failure bnet that has physical resemblance. Then we will describe a simple method of translating (i.e., mapping) any rbox diagram to a failure bnet. Then we will show how a failure bnet can be used to do all the calculations that are normally done with an rbox or an ftree diagram. In that sense, failure bnets seem to afford all the benefits of both ftree and rbox diagrams.

A **failure bnet** contains nodes of 5 types, labeled $\underline{b}$, $\underline{e}$, $\underline{x}_i$, $\underline{\phi}_i$, and $\underline{A}_i$. All nodes have only two possible states $S = Success = 0$, $F = Failure = 1$.

1. The bnet has a beginning node labeled $\underline{b}$ which is always set to success. The $\underline{b}$ node and the $\underline{\phi}_i$ nodes are the only root nodes of the bnet.

2. The bnet has a single leaf node, the end node, labeled $\underline{e}$. $\underline{e}$ is fixed. In rbox diagrams, $\underline{e} = S$ whereas in ftree diagrams, $\underline{e} = F$.

3. $\underline{x}^{nx} = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{nx-1})$. $\underline{x}_i \in \{S, F\}$ for all i.
   Suppose $\underline{x}_i$ has parents $\underline{\phi}_i$ and $\underline{a}^{na} = (\underline{a}_0, \underline{a}_1, \ldots \underline{a}_{na-1})$. Then the TPM of node $\underline{x}_i$ is defined to be

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \vee_{i=0}^{na-1} a_i) \tag{38.1}$$

4. For each node $\underline{x}_i$, the bnet has a "performance" root node $\underline{\phi}_i \in \{0, 1\}$ with an arrow pointing from it to $\underline{x}_i$ (i.e, $\underline{\phi}_i \rightarrow \underline{x}_i$). For all i,

$$P(\phi_i) = \epsilon_i \delta(\phi_i, F) + \overline{\epsilon}_i \delta(\phi_i, S) \; . \tag{38.2}$$

   ϵ_i is the failure probability and $\overline{\epsilon}_i = 1 - \epsilon_i$ is the success probability. We name the failure probability ϵ_i because it is normally very small. It is usually set to $1 - e^{-\lambda_i t} \approx \lambda_i t$ when $\lambda_i t << 1$, where λ_i is the failure rate for node $\underline{x}_i$ and t stands for time. The rblock literature usually calls $\overline{\epsilon_i} = R_i$ the **reliability** of node $\underline{x}_i$, and $\epsilon_i = (1 - R_i) = F_i$ its **unreliability**.

5. The nodes $\underline{A}_i \in \{0, 1\}$ are simply AND gates. If $\underline{A}_i$ has inputs $\underline{y}^{ny} = (\underline{y}_0, \underline{y}_1, \ldots, \underline{y}_{ny-1})$, then the TPM of $\underline{A}_i$ is

$$P(A_i|y^{ny}) = \delta(A_i, \wedge_{i=0}^{ny-1} y_i) \; . \tag{38.3}$$

An instance (instantiation) of a bnet is the bnet with all nodes set to a specific state. A **realizable instance (r-instance)** of a bnet is one which has non-zero probability.

Fig.38.3 shows how to translate any rbox diagram to a failure bnet. To illustrate this procedure, we translated the rbox diagram Fig.38.1 into the failure bnet Fig.38.4.

For the failure bnet Fig.38.4, one has:

$$\begin{array}{l} P(b) = \mathbb{1}(b = 0) \\ P(x_1|\phi_1, b) = \mathbb{1}(x_1 = \phi_1 \vee b) \\ P(x_2|\phi_2, x_1) = \mathbb{1}(x_2 = \phi_2 \vee x_1) \\ P(x_3|\phi_3, b) = \mathbb{1}(x_3 = \phi_3 \vee b) \\ P(A|x_2, x_3)e = \mathbb{1}(x_2 \wedge x_3) \\ P(e|A) = \mathbb{1}(e = A) \end{array} \quad . \tag{38.4}$$

Therefore, all r-instances of this bnet must satisfy

$$\begin{aligned} e &= (\phi_1 \vee \phi_2) \wedge \phi_3 && (38.5) \\ &= (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3) \; . && (38.6) \end{aligned}$$

Eq.(38.6) proves that Fig.38.2 is indeed a representation of Fig.38.1.

Next, we consider r-instances of this bnet for two cases: $e = S$ and $e = F$.

- **rblock analysis:** $e = S = 0$**.**
  Table 38.1 shows the probability of all possible r-instances that end in success for the failure bnet Fig.38.4. (These r-instances are the main focus of rblock analysis). The first 4 of those probabilities (those with $\phi_3 = 0$) sum to $\bar{\epsilon}_3$ so the sum $P(e = S)$ of all 5 is

  $$P(e = S) = \bar{\epsilon}_3 + \bar{\epsilon}_1\bar{\epsilon}_2\epsilon_3 \; , \tag{38.7}$$

  or, expressing it in reliability language in which $\bar{\epsilon} = R$,

  $$P(e = S) = R_3 + R_1R_2\overline{R}_3 \; . \tag{38.8}$$

- **ftree analysis:** $e = F = 1$**.**
  Table 38.2 shows the probability of all possible r-instances that end in failure for the failure bnet Fig.38.4. (These r-instances are the main focus of ftree analysis). If we set $\epsilon_i = \epsilon$ and $\bar{\epsilon}_i \approx 1$ for $i = 1, 2, 3$, then the first two of those r-instances have probabilities of $order(\epsilon^2)$ and the third has probability of $order(\epsilon^3)$. The two lowest order ($order(\epsilon^2)$) r-instances are called the "minimal cut sets" of the ftree. We will have more to say about minimal cut sets later on. For now, just note from Eq.(38.6) that the ftree Fig.38.2 is just the result of joining together with ORs two expressions, one for each of the two minimal cut sets.

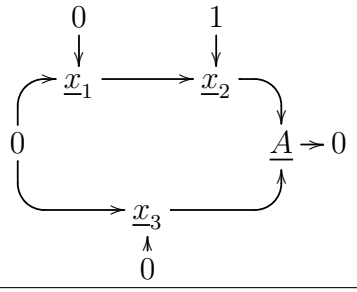
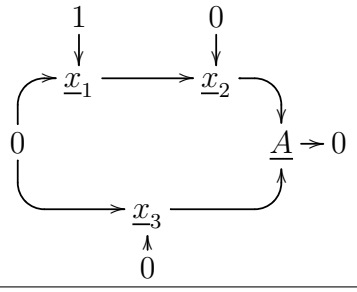
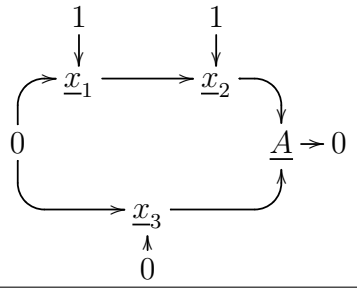
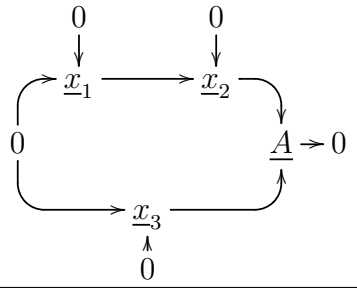
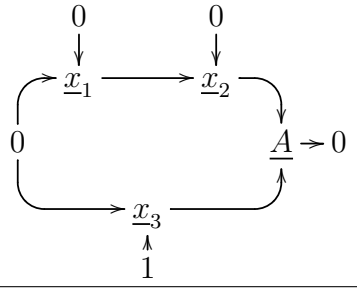
instance	probability
	$\bar{\epsilon}_1\epsilon_2\bar{\epsilon}_3$
	$\epsilon_1\bar{\epsilon}_2\bar{\epsilon}_3$
	$\epsilon_1\epsilon_2\bar{\epsilon}_3$
	$\bar{\epsilon}_1\bar{\epsilon}_2\bar{\epsilon}_3$
	$\bar{\epsilon}_1\bar{\epsilon}_2\epsilon_3$

Table 38.1: Probabilities of all possible r-instances with $e = S = 0$ for failure bnet Fig.38.4.

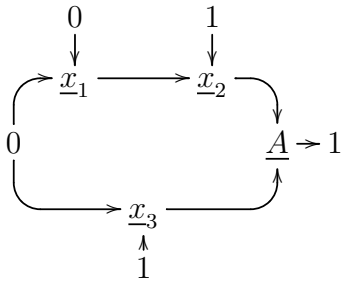
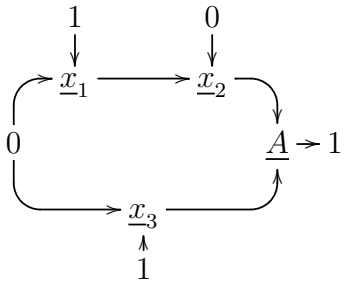
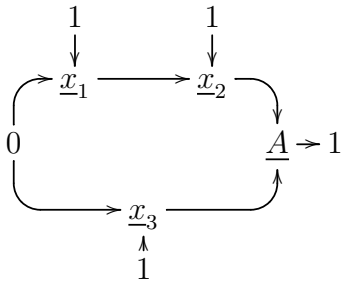
instance	probability
	$\bar{\epsilon}_1\epsilon_2\epsilon_3$
	$\epsilon_1\bar{\epsilon}_2\epsilon_3$
	$\epsilon_1\epsilon_2\epsilon_3$

Table 38.2: Probabilities of all possible r-instances with $e = F = 1$ for the failure bnet Fig.38.4.

More general $\underline{x}_i$.

Failure bnets can actually accommodate $\underline{x}_i$ nodes of a more general kind than what we first stipulated. Here are some possibilities:

For any $a^n \in \{0,1\}^n$, let

$$len(a^n) = \sum_i a_i \tag{38.9}$$

- **OR gate**

$$\begin{aligned} P(x_i|\phi_i, a^{na}) &= \delta(x_i, \phi_i \vee \vee_j a_j) && (38.10)\\ &= \delta(x_i, \phi_i \vee \mathbb{1}(len(a^{na}) > 0)) && (38.11)\end{aligned}$$

- **AND gate**

$$\begin{aligned} P(x_i|\phi_i, a^{na}) &= \delta(x_i, \phi_i \vee \wedge_j a_j) && (38.12)\\ &= \delta(x_i, \phi_i \vee \mathbb{1}(len(a^{na}) = na)) && (38.13)\end{aligned}$$

- **Fail if least K failures (less than K successes)**

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(len(a^{na}) \geq K)) \tag{38.14}$$

- **Fail if less than K failures (at least K successes)**

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(len(a^{na}) < K)) \tag{38.15}$$

- **Fail if exactly one failure**

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(len(a^{na}) = 1)) \tag{38.16}$$

  This equals an XOR (exclusive OR) gate when $na = 2$.

- **General gate**
  $f : \{0,1\}^{na} \to \{0,1\}$

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee f(a^{na})) \tag{38.17}$$

Minimal Cut Sets

Suppose $x \in \{0,1\}$ and $f : \{0,1\} \to \{0,1\}$. Then by direct evaluation, we see that

$$f(x) = [\overline{x} f(0)] \vee [x f(1)] \,. \tag{38.18}$$

Let

$$\begin{aligned} &!x = 1 - x, \\ &!^0 x = x, \\ &!^1 x = !x \end{aligned} \tag{38.19}$$

Then Eq.38.18 can be rewritten as

$$f(x) = \vee_{a \in \{0,1\}} [(!^{\overline{a}} x) f(a)] \,. \tag{38.20}$$

Now suppose $x^n \in \{0,1\}^n$ and $f : \{0,1\}^n \to \{0,1\}$. Eq.(38.20) generalizes to

$$f(x^n) = \vee_{a^n \in \{0,1\}^n} [\prod_i (!^{\overline{a}_i} x_i) f(a^n)] \,. \tag{38.21}$$

Eq.(38.21) is called an **ors-of-ands** normal form expansion. There is also an **ands-of-ors** normal form expansion obtained by swapping multiplication and $\vee$ in Eq.(38.21), but we won't need it here.

A **cut set** is a set of ϕ_i's such that if they are all equal to F, then $e = F$ for all the r-instances. A **minimal cut set** is a cut set such that there are no larger cut sets that contain it. From the failure bnet, we can always find a function $f : \{0,1\}^{nx} \to \{0,1\}$ such that $e = f(\phi^{nx})$ for all the r-instances. We did that for our example failure bnet and obtained Eq.(38.6). We can then express $f(\phi^{nx})$ as an ors-of-ands expansion to find all the minimal cut sets. The ands terms in that ors-of-ands expansion each gives a different minimal cut set, after some simplification. The ors-of-ands expression is not unique and it may be necessary to simplify (using the Boolean Algebra identities given in Chapter 0.3) to remove those redundancies.

Chapter 39

Restricted Boltzmann Machines

In what follows, we will abbreviate "restricted Boltzmann machine' by rebo.

Let
$v \in \{0,1\}^{numv}$
$h \in \{0,1\}^{numh}$
$b \in \mathbb{R}^{numv}$ (mnemonic, v and b sound the same)
$a \in \mathbb{R}^{numh}$
$W^{v|h} \in \mathbb{R}^{numv \times numh}$
Energy:

$$E(v,h) = -(b^T v + a^T h + v^T W^{v|h} h) \tag{39.1}$$

Boltzmann distribution:

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{39.2}$$

Partition function:

$$Z = \sum_{v,h} e^{-E(v,h)} = Z(a,b,W^{v|h}) \tag{39.3}$$

$$\begin{aligned} P(v|h) &= \frac{e^{b^T v + a^T h + v^T W^{v|h} h}}{\sum_v e^{b^T v + a^T h + v^T W^{v|h} h}} && (39.4)\\ &= \frac{e^{b^T v + v^T W^{v|h} h}}{\sum_v e^{b^T v + v^T W^{v|h} h}} && (39.5)\\ &= \prod_i \frac{e^{v_i (b_i + \sum_j W_{i,j}^{v|h} h_j)}}{\sum_{v_i=0,1} e^{v_i (b_i + \sum_j W_{i,j}^{v|h} h_j)}} && (39.6)\\ &= \prod_i P(v_i|h) && (39.7) \end{aligned}$$

$$P(v_i|h) = \frac{e^{v_i (b_i + \sum_j W_{i,j}^{v|h} h_j)}}{Z_i(h)} \tag{39.8}$$

Eq.39.8 implies that a rebo can be represented by the bnet Fig.39.1.

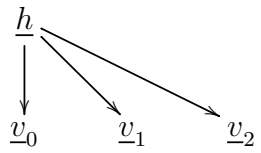


Figure 39.1: bnet for a Restricted Boltzmann Machine (rebo) with $numv = 3$

Let

$$x_i = b_i + \sum_j W_{ij}^{v|h} h_j \ . \tag{39.9}$$

Then

$$\begin{aligned} P(v_i = 1|h) &= \frac{e^{x_i}}{1 + e^{x_i}} && (39.10)\\ &= \frac{1}{1 + e^{-x_i}} && (39.11)\\ &= \text{sig}(x_i) \ . && (39.12) \end{aligned}$$

One could also expand the node $\underline{h}$ in Fig.39.1 into $numh$ nodes. But note that $P(h) \neq \prod_j P(h_j)$ so there would be arrows among the h_j nodes.

Note that the rebo bnet is a special case of Naive Bayes (See Chapter 30) with $v_i, h_i \in \{0, 1\}$ and specific $P(h)$ and $P(v_i|h)$ node matrices.

Chapter 40

Scoring the Nodes of a Learned Bnet

Chapter 42 discusses how to learn a bnet from data. Many algorithms for doing this require scoring how well a particular bnet fits the data. This chapter is an introduction to such scoring.

Normally, each node of a bnet is scored separately, and then those node scores are summed to get the bnet score.

In this chapter, scores are defined so that a higher score means a better fit. By taking the negative of such a score, one can always get a score such that a lower score means a better fit.

There are 2 main types of bnet scores: Maximum Likelihood (ML) scores, and Shannon Information Theory (SIT) scores. ML scores consist of the log of a maximum likelihood function $P(\vec{x}|\theta)$ for i.i.d. samples $\vec{x} = (x[\sigma])_{\sigma=0,1,\ldots,nsam-1}$, where $x[\sigma] \sim P_{\underline{x}|\theta}(x|\theta)$:

$$\begin{aligned}
\text{ML-score} &= \ln(P(\vec{x}|\theta)) && (40.1)\\
&= \ln \prod_{\sigma} P(x[\sigma] \mid \theta) && (40.2)\\
&= \sum_{\sigma} \ln P(x[\sigma] \mid \theta) && (40.3)\\
&\approx nsam \sum_{x} P(x|\theta) \ln P(x|\theta) && (40.4)\\
&= -nsamH(P_{\underline{x}|\theta}) \ , && (40.5)
\end{aligned}$$

and SIT scores consist of a negative entropy:

$$\text{Info-score} = -H(P_{\underline{x}|\theta}) \ . \qquad (40.6)$$

Thus, up to a factor of $nsam$, they are the same thing. Maximizing a log likelihood function for i.i.d. samples or minimizing the corresponding entropy, are the same thing, and they both yield a good estimate of the hidden parameters θ.

Probability Distributions and Special Functions

While writing this chapter, I briefly consulted the following Wikipedia articles about the definitions and properties of certain probability distributions and special functions.

- Categorical Distribution, Ref.[36]
- Multinomial Distribution, Ref.[55]
- Dirichlet Distribution, Ref.[39]
- Multivariate Normal Distribution, Ref.[57]
- Beta function, Ref.[33]
- Multinomial Coefficients, Ref.[56]
- Gamma Function Ref.[41]

Here are a few results from those Wikipedia articles that we will use later on in this chapter. Below, we will abbreviate $q_+ = \sum_i q_i$, and $q_\cdot = (q_0, q_1, \ldots, q_{nq-1})$ for various quantities q

Gamma function. If $n > 0$ is an integer,

$$\Gamma(n+1) = n! \tag{40.7}$$

The **multivariate Beta function** is defined by

$$B(\alpha_\cdot) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\alpha_+)} \tag{40.8}$$

where $\alpha_k > 0$ for all k.

The **multinomial coefficient** is defined by

$$C(N_\cdot) = \frac{N_+!}{\prod_k N_k!} \tag{40.9}$$

where N_k are non-negative integers.

The **inverse of the multinomial coefficient** will be denoted by

$$CI(N_\cdot) = \frac{1}{C(N_\cdot)} = \frac{\prod_k N_k!}{N_+!} \tag{40.10}$$

The **Categorical Distribution** is defined by

$$Cat(x; \pi_\cdot) = \pi_x = \prod_k \pi_k^{\mathbb{1}(k=x)} \tag{40.11}$$

for $k, x \in S_{\underline{x}}$, where $\pi_\cdot$ is a probability dist.(i.e., $\pi_k \geq 0$ for all k, and $\pi_+ = 1$).

The **Multinomial Distribution** is defined by

$$Mul(N.; \pi., N) = C(N.) \ , \tag{40.12}$$

where N_k is a non-negative integer for all k, $N_+ = N$, and $\pi.$ is a probability dist. Mul() satisfies:

$$E[\underline{N_k}] = N\pi_k \ . \tag{40.13}$$

The **Dirichlet Distribution** is defined by

$$Dir(\pi.; \alpha.) = \frac{1}{B(\alpha.)} \prod_k \pi_k^{\alpha_k - 1} \tag{40.14}$$

where $\alpha_k > 0$ for all k, and $\pi.$ is a probability dist. The $\alpha.$ are called **concentration parameters** or **hyperparameters**. Dir() satisfies:

$$E[\underline{\pi_k}] = \frac{N_k}{N_+} \ . \tag{40.15}$$

Dir() is **conjugate prior** of Mul()

Note that

$$Mul(N.; \pi., N) Dir(\pi.; \alpha.) = \mathcal{K}(N., \alpha.) Dir(\pi.; N. + \alpha.) \ , \tag{40.16}$$

where

$$\mathcal{K}(N., \alpha.) = \frac{B(N. + \alpha.)}{CI(N.) B(\alpha.)} \ . \tag{40.17}$$

Dir() is replaceable by a Mul() for large concentration parameters

Note that if N_k is a positive integer and $\alpha_k = N_k + 1$ for all k, then

$$\begin{aligned} Dir(\pi.; \alpha_k = N_k + 1) &= C(N.) \prod_k \pi_k^{N_k} & (40.18) \\ &= Mul(N.; \pi., N_+) \ . & (40.19) \end{aligned}$$

Single node with no parents

In this section, we consider a learned bnet consisting of a single node with no parents. We will consider arbitrary learned bnets in the next section. But we start with this simplified case so as to reduce the number of indices in most quantities from 3 to 1. All the results that we derive in this section will be used in the next section after adding the extra indices. This way, we will avoid carrying the extra indices throughout the intermediate steps of many derivations.

For state $k \in \{0, 1, \ldots, nk - 1\}$ of a single node $\underline{x}$, let

$\underline{N_k}$= current count number (an integer, data)

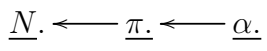


Figure 40.1: For a bnet consisting of a single node with no parents, this is a Markov chain of current counts ($\underline{N}.$), TPM ($\underline{\pi}.$), and prior counts ($\underline{\alpha}.$) .

$\underline{\pi}.$= a probability dist, the TPM for the node
$\underline{\alpha}_k$= prior count number
Consider the Markov chain bnet of Fig.40.1, with the following TPMs, given in blue.

$$P(N.|\pi.) = Mul(N.; \pi., N_+) \tag{40.20}$$

$$P(\pi.|\alpha.) = Dir(\pi.; \alpha.) \tag{40.21}$$

It follows that

$$\begin{aligned} P(N., \pi.|\alpha.) &= P(N.|\pi.)P(\pi.|\alpha.) && (40.22)\\ &= Mul(N.; \pi., N_+)Dir(\pi.; \alpha.) && (40.23)\\ &= \mathcal{K}(N., \alpha.)Dir(\pi.; N. + \alpha.) \ . && (40.24) \end{aligned}$$

From Eq.(40.15) for the expected value of Dir(), we get

$$\hat{\pi}. = E[\underline{\pi}.] = \frac{N. + \alpha.}{N_+ + \alpha_+} \ . \tag{40.25}$$

Integrating both sides of Eq.(40.24) over $\pi.$, we find that

$$P(N.|\alpha.) = \mathcal{K}(N., \alpha.) \ . \tag{40.26}$$

If $N_k >> 1$ for all k, then the Dir() in Eq.(40.24) can be replaced by a Mul()

$$P(N., \pi.|\alpha.) \approx \mathcal{K}(N., \alpha.)Mul(N. + \alpha.; \pi., N_+ + \alpha_+) \ . \tag{40.27}$$

Therefore,

$$\begin{aligned} P(N.|\pi., \alpha.) &= \frac{P(N., \pi.|\alpha.)}{P(N.|\alpha.)} && (40.28)\\ &= Mul(N. + \alpha.; \pi., N_+ + \alpha_+) \ . && (40.29) \end{aligned}$$

Claim 23

$$\begin{aligned} \ln P(N.|\hat{\pi}., \alpha.) &= -(N_+ + \alpha_+)H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) + \ln C(N. + \alpha.) && (40.30)\\ &> -(N_+ + \alpha_+)H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) - \frac{1}{2}(nk - 1)\ln N_+ && (40.31) \end{aligned}$$

proof:

$$\begin{aligned}
\ln P(N.|\hat{\pi}., \alpha.) &= \sum_k (N_k + \alpha_k) \ln \hat{\pi}_k + \ln C(N. + \alpha.) && (40.32)\\
&= \sum_k (N_k + \alpha_k) \ln \frac{N_k + \alpha_k}{N_+ + \alpha_+} + \ln C(N. + \alpha.) && (40.33)\\
&= -(N_+ + \alpha_+) H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) + \ln C(N. + \alpha.) && (40.34)
\end{aligned}$$

Recall Stirling's approximation of a factorial, valid for large integers n:

$$\ln n! \approx (n + \frac{1}{2}) \ln n - n \ . \tag{40.35}$$

Assume $N_k >> 1$ for all k. Applying Stirling's approximation to all factorials in $C(N)$, we get

$$\begin{aligned}
\ln C(N.) &\approx (N_+ + \frac{1}{2}) \ln N_+ - N_+ - \sum_k \left[(N_k + \frac{1}{2}) \ln N_k - N_k \right] && (40.36)\\
&= (N_+ + \frac{1}{2}) \ln N_+ - \sum_k (N_k + \frac{1}{2}) \ln N_k \ . && (40.37)
\end{aligned}$$

Next assume that

$$N_k \approx \frac{N_+}{nk} \ . \tag{40.38}$$

Then

$$\begin{aligned}
\ln C(N.) &= (N_+ + \frac{1}{2}) \ln N_+ - nk(\frac{N_+}{nk} + \frac{1}{2})[\ln N_+ - \ln nk] && (40.39)\\
&= -\frac{1}{2}(nk - 1) \ln N_+ + (N_+ + \frac{nk}{2}) \ln nk && (40.40)\\
&> -\frac{1}{2}(nk - 1) \ln N_+ \ . && (40.41)
\end{aligned}$$

QED

Multiple nodes with any number of parents

In the previous section, we considered a bnet consisting of a single node with no parents, so we only needed a single index k for the states of the single node. In this section, we consider an arbitrary bnet with multiple nodes each of which may have multiple parents. Most of the results in the

$$\underline{N^i_{\cdot,\mu}} \longleftarrow \underline{\pi^i_{\cdot|\mu}} \longleftarrow \underline{\alpha^i_{\cdot,\mu}}$$

Figure 40.2: Generalization of Fig.40.1. For a bnet with multiple nodes each of which may have multiple parents, this is a Markov chain of current counts ($\underline{N^i_{\cdot,\mu}}$), TPM ($\underline{\pi^i_{\cdot|\mu}}$), and prior counts ($\underline{\alpha^i_{\cdot,\mu}}$) .

previous section are valid for the general case if we make the following replacements: $\pi. \to \pi^i_{\cdot|\mu}$ $N. \to N^i_{\cdot,\mu}$ $\alpha. \to \alpha^i_{\cdot,\mu}$. Upon this replacement, Fig.40.1 becomes Fig.40.2. The TPMs, printed in blue, of the new Markov chain, are as follows:

$$P(N^i_{\cdot,\mu}|\pi^i_{\cdot|\mu}) = Mul(N^i_{\cdot,\mu}; \pi^i_{\cdot|\mu}, N^i_{+,\mu}) \tag{40.42}$$

$$P(\pi^i_{\cdot|\mu}|\alpha^i_{\cdot,\mu}) = Dir(\pi^i_{\cdot|\mu}; \alpha^i_{\cdot,\mu}) \tag{40.43}$$

In these TPMs,
$i \in S_{\underline{i}} = \{0, 1, \ldots, ni-1\}$= node index
$\underline{x}. = (\underline{x}_i)_{i\in S_{\underline{i}}}$= the nodes of the learned bnet.
$k \in S_{\underline{k}^i} = \{0, 1, \ldots, nk^i - 1\}$= states of node $\underline{x}_i$
$\mu \in S_{\underline{\mu}^i} = \{0, 1, \ldots, n\mu^i - 1\}$= states of parents of node $\underline{x}_i$.

In the previous section, we assumed a single node ($ni = 1$) with no parents ($n\mu^0 = 1$) so that we could drop the i, μ indices. In this section, we eliminate that restriction.

It is convenient to define the magnitude of a bnet B to equal the sum over nodes of the number of free parameters in each TPM:

$$|B| = \sum_i (nk^i - 1)n\mu^i \ . \tag{40.44}$$

Suppose that we are given $nsam$ samples $\vec{\underline{x}}_i = (\underline{x}_i[\sigma])_{\sigma=0,1,\ldots,nsam-1}$ of our learned bnet. The count numbers $N^i_{k,\mu}$ are defined in terms of those samples as follows:

$$N^i_{k,\mu} = \sum_\sigma \mathbb{1}(\underline{x}_i[\sigma] = k, pa(\underline{x}_i[\sigma]) = \mu) \ . \tag{40.45}$$

It is also convenient to defined count number ratios

$$N^i_{k|\mu} = \frac{N^i_{k,\mu}}{N^i_{+,\mu}} \ . \tag{40.46}$$

Note that $N^i_{k,\mu}$ is a positive integer whereas $N^i_{k|\mu} \in [0, 1]$.

Let's denote the components of the TPMs by $\pi^i_{k|\mu}$:

$$\pi^i_{k|\mu} = P(\underline{x}_i = k \mid pa(\underline{x}_i) = \mu) \approx N^i_{k|\mu} \ . \tag{40.47}$$

The rest of this section lists equations that we obtained from the previous section, by adding the new indices i, μ:

$$\mathcal{K}(N^i_{\cdot,\mu}, \alpha^i_{\cdot,\mu}) = \frac{B(N^i_{\cdot,\mu} + \alpha^i_{\cdot,\mu})}{CI(N^i_{\cdot,\mu})B(\alpha^i_{\cdot,\mu})} \tag{40.48}$$

$$\hat{\pi}^i_{k|\mu} = \frac{N^i_{k,\mu} + \alpha^i_{k,\mu}}{N^i_{+,\mu} + \alpha^i_{+,\mu}} \tag{40.49}$$

$$P(N^i_{\cdot,\mu}|\alpha^i_{\cdot,\mu}) = \mathcal{K}(N^i_{\cdot,\mu}, \alpha^i_{\cdot,\mu}) \tag{40.50}$$

$$P(N^i_{\cdot,\mu}|\pi^i_{\cdot|\mu}, \alpha^i_{\cdot,\mu}) \approx Mul(N^i_{\cdot,\mu} + \alpha^i_{\cdot,\mu}; \pi^i_{\cdot|\mu}, N^i_{+,\mu} + \alpha^i_{+,\mu}) \tag{40.51}$$

Claim 24

$$\ln P(N^i_{\cdot,\mu}|\hat{\pi}^i_{\cdot|\mu}, \alpha^i_{\cdot,\mu}) = \sum_k (N^i_{k,\mu} + \alpha^i_{k,\mu}) \ln \left(\frac{N^i_{k,\mu} + \alpha^i_{k,\mu}}{N^i_{+,\mu} + \alpha^i_{+,\mu}} \right) + \ln C(N^i_{\cdot,\mu} + \alpha^i_{\cdot,\mu}) \tag{40.52}$$

$$> \sum_k (N^i_{k,\mu} + \alpha^i_{k,\mu}) \ln \left(\frac{N^i_{k,\mu} + \alpha^i_{k,\mu}}{N^i_{+,\mu} + \alpha^i_{+,\mu}} \right) - \frac{1}{2}(nk^i - 1) \ln N^i_{+,\mu} \tag{40.53}$$

Bayesian Scores

- Bayesian Information Criterion (BIC)

$$\text{BIC-score} = -\sum_i \sum_{k,\mu} N^i_{k,\mu} \ln \left(\frac{N^i_{k,\mu}}{N^i_{+,\mu}} \right) + \underbrace{\left[-\frac{|B|}{2} \ln N^+_{+,+} \right]}_{\sum_i \sum_\mu \ln C(N^i_{\cdot,\mu}) \text{ would be more accurate}} \tag{40.54}$$

$$\approx \sum_i \sum_\mu \ln P(N^i_{\cdot,\mu}|\hat{\pi}^i_{\cdot|\mu}, \alpha^i_{\cdot,\mu} = 0) \tag{40.55}$$

- Bayesian Dirichlet (BD)

$$\text{BD-score} = \sum_i \sum_\mu \ln \frac{B(N^i_{\cdot,\mu} + \alpha^i_{\cdot,\mu})}{B(N^i_{\cdot,\mu})} \tag{40.56}$$

$$= \sum_i \sum_\mu \ln \left[CI(N^i_{\cdot,\mu}) P(N^i_{\cdot,\mu}|\alpha^i_{\cdot,\mu}) \right] \tag{40.57}$$

- BD equivalent (BDe)

$$\text{BDe-score} \quad = \quad \text{BD-score}\left(\alpha^i_{k,\mu} = \alpha' N^i_{k,\mu}\right) \ , \tag{40.58}$$

  where α' is a free parameter.

- BD equivalent unified (BDeu)

$$\text{BDeu-score} \quad = \quad \text{BD-score}\left(\alpha^i_{k,\mu} = \frac{\alpha'}{nk^i n\mu^i}\right) \ , \tag{40.59}$$

  where α' is a free parameter. The BDeu score satisfies **score equivalence**; i.e., it is the same for all DAGs in an equivalence class of observational equivalent DAGs. See Chapter 34 for more information about observational equivalence.

Information Theoretic scores

- Maximum likelihood

$$\text{ML-score} \quad = \quad \sum_i \sum_{k,\mu} N^i_{k,\mu} \ln N^i_{k|\mu} \tag{40.60}$$

$$= \quad -\sum_i H(\underline{k}^i | \underline{\mu}^i) \ , \tag{40.61}$$

  where $P_{\underline{k}^i|\underline{\mu}^i}(k|\mu) = N^i_{k|\mu}$ and $P_{\underline{k}^i,\underline{\mu}^i}(k,\mu) = N^i_{k,\mu}$.

- Bayesian Information Criterion (BIC), aka Minimum Description Length (MDL)

$$\text{BIC-score} \quad = \quad \text{ML-score} - \frac{|B|}{2} \ln N^+_{+,+} \tag{40.62}$$

$$\approx \quad \sum_i \sum_{k,\mu} N^i_{k,\mu} \ln \frac{N^i_{k|\mu}}{\sqrt{N^+_{+,+}}} \tag{40.63}$$

- Akaike Information Criterion (AIC)

$$\text{AIC-score} \quad = \quad \text{ML-score} - |B| \tag{40.64}$$

$$\approx \quad \sum_i \sum_{k,\mu} N^i_{k,\mu} \left[\ln N^i_{k|\mu} - 1\right] \tag{40.65}$$

Chapter 41

Simpson's Paradox

This chapter is based on Chapter 6 of "The Book of Why", Ref.[22]. See also Ref.[61] and references therein.

Simpson's paradox is a recurring nightmare for all statisticians overseeing a clinical trial for a medicine. It is possible that if they leave out a certain "confounding" variable from a study, the study's conclusion on whether a medicine is effective or not, might be, without measuring that confounding variable, the opposite of what it would have been had that variable been measured.

Simpson's Paradox is greatly clarified by Judea Pearl's theory of causality. At the end of this chapter, we explain how.

Here is a simple example of Simpson's Paradox.

An equal number of patients of male and female genders are given a heart medicine or a placebo in a double blind study. Some subsequently have a heart attack. Let

$\underline{a}$ = heart attack? No=0, Yes=1

$\underline{t}$ = took medicine? No=0, Yes=1

$\underline{g}$ = gender? Female=0, Male=1

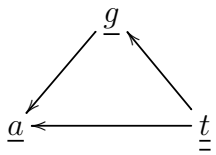


Figure 41.1: bnet for a simple example of Simpson's paradox. Here node $\underline{g}$ is a chain junction and a mediator.

This situation can be modeled by either bnet Fig.41.1. or bnet Fig.41.2. The two bnets are probabilistically equivalent (i.e., they both represent the same probability distribution $P(a,t,g)$) because

$$P(g|t)P(t) = P(g,t) = P(t|g)P(g)\ . \tag{41.1}$$

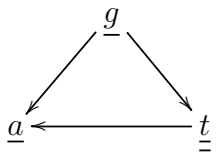


Figure 41.2: bnet that is probabilistically but not physically equivalent to bnet Fig.41.1. Here node $\underline{g}$ is a fork junction and a confounder.

For the bnet Fig.41.1, one has

$$P(a,g,t) = P(a|g,t)P(g|t)P(t) \ . \tag{41.2}$$

Therefore,

$$P(a=1|t) = \sum_g P(a=1|t,g)P(g|t) = E_{\underline{g}|t}P(a=1|t,\underline{g}) \ , \tag{41.3}$$

where $E_{\underline{g}|t}$ is a conditional expected value (a kind of weighted average).

Suppose q_0, q_1 are non-negative real numbers. For the vector $\vec{q} = (q_0, q_1)$:
Define a negative outcome (or failure or q_t increasing with t) if $q_0 \leq q_1$.
Define a positive outcome (or success or q_t decreasing with t) if $q_0 \geq q_1$.
Let

$$\vec{q}^{\,g} = [P(\underline{a}=1|t,g)]_{t=0,1} \tag{41.4}$$

for $g = 0, 1$, and

$$\vec{q}^{\,*} = [P(\underline{a}=1|t)]_{t=0,1} \ . \tag{41.5}$$

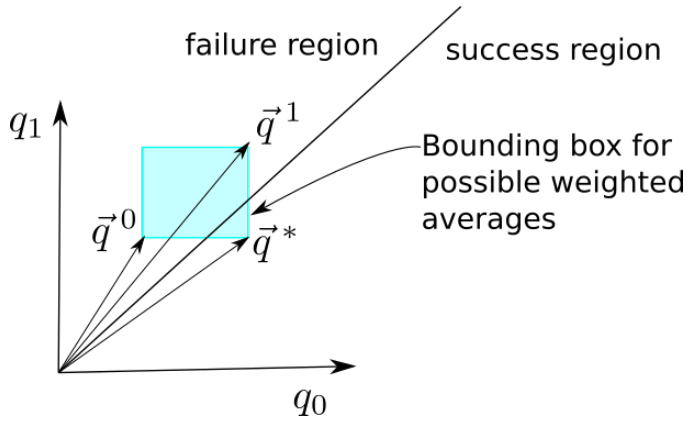


Figure 41.3: $\vec{q}^{\,0}$, $\vec{q}^{\,1}$ vectors and bounding box for vector $\vec{q}^{\,*}$.

It is possible (see Fig.41.3 for a graphical explanation of how) to find perverse cases in which $P(a = 1|t, g = 0)$ and $P(a = 1|t, g = 1)$ increase with t but $P(a = 1|t)$ decreases with t. So it is possible to conclude that the medicine is a failure for each of the two g populations considered separately, yet the medicine is a success when both populations are "amalgamated". The lesson is that a "trend reversal" is possible upon amalgamation. Trends are not necessarily preserved when we do a weighted average of type $E_{\underline{g}|t}$. $E_{\underline{g}|t}$ is an expected value on the random variable $\underline{g}$ conditioned on the root random variable $\underline{t}$.

So far, we have proven that probabilistically, the drug can be a failure for the populations of both sexes considered separately, but a success for the aggregate population.

Pearl Causality

Pearl Causality would add the following two important insights to this problem:

1. bnets Fig.41.1 and Fig.41.2, although they are probabilistically equivalent, do not represent the same physical situation. In fact, only Fig.41.2 occurs in this case.

2. To decide whether the medicine is effective, we must apply a $do()$ operator to the $\underline{t}$ variable in Fig.41.2. The effect of that $do()$ operator is to erase the arrow going from $\underline{g}$ to $\underline{t}$. This in turn means that the average $E_{\underline{g}|t}$ in our equation for $P(a = 1|t)$ becomes a simpler average $E_{\underline{g}}$ which is independent of $\underline{t}$. But for such an average, the bounding box in Fig.41.3 degenerates to its diagonal line that connects the tips of the two vectors $\vec{q}^{\,0}$ and $\vec{q}^{\,1}$. The vector $\vec{q}^{\,*}$ must now fall on that diagonal line and must therefore also fall in the success region.

In conclusion, as Judea Pearl would say, if we ask the right question to Nature, i.e., what is $P[a = 1|do(\underline{t} = t)]$ for $t = 0, 1$, we get as an answer that the aggregate population preserves rather than reverses the unanimous trend of the two gendered populations.

Numerical Example

(a,t,g)	number of patients segregated by gender	number of patients of either gender
0,0,0	19	47
0,0,1	28	
0,1,0	37	49
0,1,1	12	
1,0,0	1	13
1,0,1	12	
1,1,0	3	11
1,1,1	8	

Table 41.1: Data for numerical example of Simpson's Paradox. This fictitious data was taken directly from Table 6.4, page 210 of "The Book of Why", Ref.[22].

$$P(a|t,g) = \begin{array}{c|cccc} & 0,0 & 0,1 & 1,0 & 1,1 \\ \hline 0 & 19/20 & 28/40 & 37/40 & 12/20 \\ 1 & 1/20 & 12/40 & 3/40 & 8/20 \end{array} \tag{41.6}$$

$$P(a|t) = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 47/60 & 49/60 \\ 1 & 13/60 & 11/60 \end{array} \tag{41.7}$$

$$\begin{aligned} \frac{P(a=1,t=1,g=0)}{\sum_a P(a,t=1,g=0)} &= P(a=1|t=1,g=0) &&= \frac{3}{40} \\ \frac{P(a=1,t=0,g=0)}{\sum_a P(a,t=0,g=0)} &= P(a=1|t=0,g=0) &&= \frac{1}{20} = \frac{2}{40} \end{aligned} \tag{41.8}$$

$$\begin{aligned} \frac{P(a=1,t=1,g=1)}{\sum_a P(a,t=1,g=1)} &= P(a=1|t=1,g=1) &&= \frac{8}{20} = \frac{16}{40} \\ \frac{P(a=1,t=0,g=1)}{\sum_a P(a,t=0,g=1)} &= P(a=1|t=0,g=1) &&= \frac{12}{40} \end{aligned} \tag{41.9}$$

$$\begin{aligned} \frac{\sum_g P(a=1,t=1,g)}{\sum_g \sum_a P(a,t=1,g)} &= P(a=1|t=1) &&= \frac{11}{60} \\ \frac{\sum_g P(a=1,t=0,g)}{\sum_g \sum_a P(a,t=0,g)} &= P(a=1|t=0) &&= \frac{13}{60} \end{aligned} \tag{41.10}$$

Note that the right hand side of Eq.41.8 is higher for $t = 1$ than for $t = 0$. Same trend occurs in Eqs.41.9 but is reversed in Eqs.41.10.

Chapter 42

Structure and Parameter Learning for Bnets

Learning a bnet from data is a computationally intensive NP-complete problem. Therefore, the best one can hope for is for heuristic algorithms that solve this problem approximately. A huge number of such algorithms have been tried and continue to be tried. Luckily, there exists a free open source software library called `bnlearn` that covers many of them. The goal of this chapter is to give a brief overview of the subject of bnet learning, after which we recommend to those readers who want to pursue this subject further, to learn `bnlearn` .

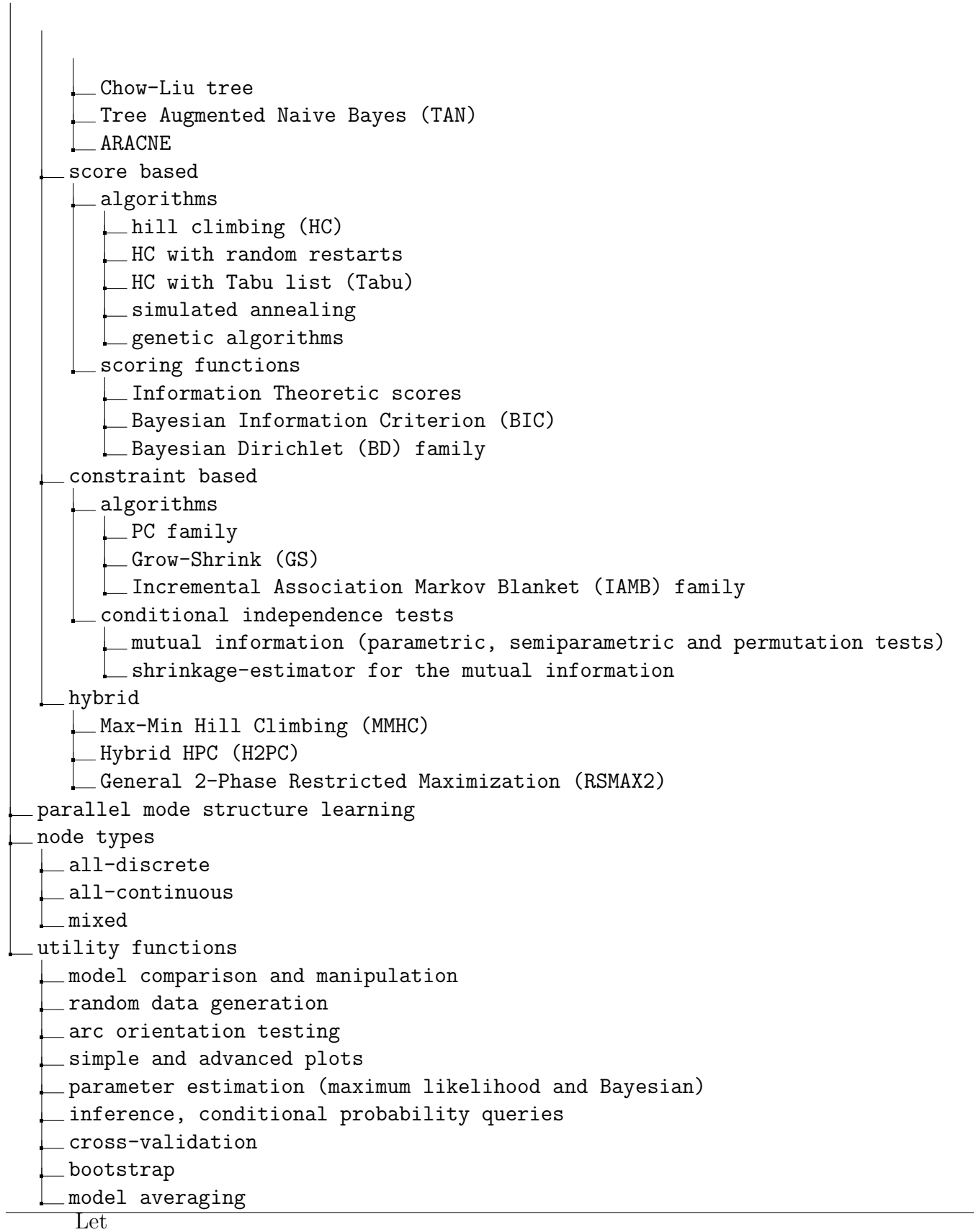
This chapter is based on the `bnlearn` website Ref.[24], and on a 2019 survey paper [25] by Scutari et al. I highly recommend looking at both. Refs. [2] and [10] were also helpful to me in understanding this subject.

`bnlearn` (Ref.[24]) (free, open source) is very comprehensive and well maintained. It is written mostly in C with an R frontend. It was developed by Marco Scutari and collaborators over a time period of more than 10 years, and is still under active development. How things stand in the field of bnet learning software reminds me of how things stand in the field of linear algebra (LA) software. Perfecting and optimizing LA software takes many years so I would not advise you to write your own LA software library starting from scratch. There is no need to do so. Instead, you can use LAPACK (free, open source), which has been perfected and expanded for decades by world experts. I view `bnlearn` as the LAPACK of bnet learning.

Overview

To give the reader an overview of the subject and of `bnlearn` itself, here is a highly simplified tree, compiled from the `bnlearn` website and documentation, of some of the subjects covered by `bnlearn` .

```
Parameter Learning
   missing data
Structure Learning
   tree-like structures given a priori
      Naive Bayes
```


Chow-Liu tree
Tree Augmented Naive Bayes (TAN)
ARACNE
score based
algorithms
hill climbing (HC)
HC with random restarts
HC with Tabu list (Tabu)
simulated annealing
genetic algorithms
scoring functions
Information Theoretic scores
Bayesian Information Criterion (BIC)
Bayesian Dirichlet (BD) family
constraint based
algorithms
PC family
Grow-Shrink (GS)
Incremental Association Markov Blanket (IAMB) family
conditional independence tests
mutual information (parametric, semiparametric and permutation tests)
shrinkage-estimator for the mutual information
hybrid
Max-Min Hill Climbing (MMHC)
Hybrid HPC (H2PC)
General 2-Phase Restricted Maximization (RSMAX2)
parallel mode structure learning
node types
all-discrete
all-continuous
mixed
utility functions
model comparison and manipulation
random data generation
arc orientation testing
simple and advanced plots
parameter estimation (maximum likelihood and Bayesian)
inference, conditional probability queries
cross-validation
bootstrap
model averaging

Let

- PL=parameters learning (i.e, learning the TPMs)
- SL= structure learning (i.e.,learning the DAG)

PL is easy, once the structure is known. PL assuming no missing data goes as follows. Using the notation of Chapter 40, define

$$\pi^i_{k|\mu} = P(\underline{x}_i = k \mid pa(\underline{x}_i) = \mu) \ . \tag{42.1}$$

Then $\pi^i_{k|\mu}$ can be estimated from the data $N^i_{k,\mu}$ using:

$$\pi^i_{k|\mu} \approx N^i_{k|\mu} = \frac{N^i_{k,\mu}}{N^i_{+,\mu}} \ . \tag{42.2}$$

PL described by Eq.(42.2) is only for discrete nodes with no missing data. `bnlearn` can also do PL with missing data and continuous (Gaussian linear only) nodes. See Chapter 28 on missing data and Chapter 17 on Gaussian linear nodes. SL actually does PL and SL at the same time.

There are 3 main types of SL: score based, constraint based, and hybrid. `bnlearn` can perform many algorithms of each of these 3 types of SL. It can perform most of them with either all-discrete, or all-continuous or mixed nodes. It can perform many of them in parallel mode. The 2019 survey paper Ref.[25] by Scutari et al compares the performance of many different bnet learning algorithms.

Score based SL algorithms

Score based SL algorithms require scoring bnets (with either all-discrete, all-continuous or mixed nodes). See Chapter 40 for an introduction to scoring bnets. The BIC score explained in that chapter is very popular and works for all-discrete, all-continuous or mixed nodes.

Score-based SL algorithms apply standard optimisation techniques. In the Hill Climbing algorithm, the current best bnet is changed slightly and then given a score that measures how well it fits the data. The bnet with the highest (=best) score so far, as well as that highest score, are stored. (Hence, this is called a greedy search). The process continues until the latest highest score stops changing. The problem with being greedy all the time is that the answer might converge to a local maximum. To mitigate this problem and allow some probability of visiting more than one local maximum, one uses a Tabu Table, random restarts, simulated annealing, genetic algorithms, etc.

Constraint based SL algorithms

To fully understand constraint based SL algorithms, the reader is advised to read Chapters 12 and 34 first.

Constraint based SL algorithms require estimating from the data the conditional independence $\underline{x}. \perp_P \underline{y}.|\underline{a}.$ for any 3 disjoint multinodes $\underline{x}., \underline{y}., \underline{a}.$. This can be done by estimating the conditional mutual information (CMI) $H(\underline{x}. : \underline{y}.|\underline{a}.)$. `bnlearn` can calculate CMI and other metrics of $\underline{x}. \perp_P \underline{y}.|\underline{a}.$. All these metrics are very similar; they all measure how close $P(x.|y., a.)$ and $P(x.|a.)$ are.

The first constraint-based SL algorithm was the Inductive Causation (IC) algorithm proposed by Pearl and Verma in 1991. Incremental improvements have been proposed since then, such as the PC family of algorithms, Grow-Shrink and the Incremental Association Markov Blanket (IAMB) family of algorithms.

Pseudo-code for some bnet learning algorithms

Algorithm 1: Pseudo-code for Hill Climbing algorithm

Input : Data D, Vertices V

Output: a bnet $B = (G, T)$, where $G = (V, E)$ is a DAG, where V are its vertices (nodes) and E are its edges (arrows). T are all its Transition Probability Matrices (TPMs) $T = TPMs(G, D)$.

$E \leftarrow \varnothing$
$T \leftarrow \varnothing$
$B \leftarrow (V, E, T)$
$maxscore \leftarrow -\infty$
// DE= all possible directed edges
$DE = \{\underline{x} \rightarrow \underline{y} \in V \times V : \underline{x} \neq \underline{y}\}$
$again \leftarrow True$
while *again* **do**
  for *all* $\underline{x} \rightarrow \underline{y} \in DE$ **do**
    // add arrow
    $E_+ \leftarrow E \cup \{\underline{x} \rightarrow \underline{y}\}$
    // delete arrow
    $E_- \leftarrow E - \{\underline{x} \rightarrow \underline{y}\}$
    // reverse arrow
    $E_R \leftarrow E_- \cup \{\underline{y} \rightarrow \underline{x}\}$
    for $E' = E_+, E_-, E_R$ **do**
      if $E' \neq E$ *and* $G' = (V, E')$ *is a legal DAG* **then**
        $T' \leftarrow TPMs(G', D)$
        $B' \leftarrow (G', T')$
        $newscore$ = BIC-score(B')
        if $newscore > maxscore$ **then**
          $B \leftarrow B'$
          $maxscore \leftarrow newscore$
        else
          $again \leftarrow False$
return B

Algorithm 2: Pseudo-code for PC-Stable algorithm

Input : Data D, Vertices (nodes) V, tolerance in CMI $\epsilon > 0$

Output: partially oriented acyclic graph $G = (V, E, UE)$, where V are the vertices (nodes), E are the oriented edges (arrows) and UE are the unoriented edges.

$E \leftarrow \varnothing$

// initialize UE to fully-connected undirected graph

$UE \leftarrow \{\underline{x} - \underline{y} \in V \times V : \underline{x} - \underline{y} = \underline{y} - \underline{x}, \underline{x} \neq \underline{y}\}$

// Shrink phase. Deletes edges from E.

for $\lambda = 0, 1, 2, \ldots, |V| - 2$ **do**

  for *all* $\underline{x} - \underline{y} \in UE$ **do**

    for *all* $S = \{\underline{a} \in V : \underline{x} - \underline{a} \in UE, \underline{a} \neq \underline{x}, \underline{y}\} \ni |S| = \lambda$ **do**

      if $H(\underline{x} : \underline{y}|S) < \epsilon$ **then**

        /* If there were an arrow between $\underline{x}$ and $\underline{y}$, then conditioning on S would not be enough to interrupt info transmission $H(\underline{x} : \underline{y}|S)$ between $\underline{x}$ and $\underline{y}$ */

        $UE \leftarrow UE - \{\underline{x} - \underline{y}\}$

        $S(\underline{x} - \underline{y}) \leftarrow S$

// Growth phase. Adds v structures to E.

for *all* $\underline{x}, \underline{y}, \underline{a}$ *such that* $\underline{x} - \underline{a} \in UE, \underline{a} - \underline{y} \in UE, \underline{x} - \underline{y} \notin UE, \underline{a} \notin S(\underline{x} - \underline{y})$ **do**

  /* If there were no collider at $\underline{a}$, then there would be info transmission between $\underline{x}$ and $\underline{y}$ */

  $UE \leftarrow UE - \{\underline{x} - \underline{a}, \underline{a} - \underline{y}\}$

  $E \leftarrow E \cup \{\underline{x} \rightarrow \underline{a}, \underline{y} \rightarrow \underline{a}\}$

// Orienting edges.

$again \leftarrow True$

$size \leftarrow |UE|$

while *again* **do**

  for *all* $\underline{x} - \underline{y} \in UE$ **do**

    if $\underline{x} \rightarrow \underline{y} \in E$, $\underline{y} - \underline{z} \in UE$, $\underline{x} - \underline{z} \notin UE$, $\nexists\, \underline{w} \ni \underline{w} \rightarrow \underline{y} \in E$ **then**

      // to avoid introducing new v structure

      $UE \leftarrow UE - \{\underline{y} - \underline{z}\}$

      $E \leftarrow E \cup \{\underline{y} \rightarrow \underline{z}\}$

    if $\underline{x} \rightarrow \underline{y} \in E$ *and there is directed path from* $\underline{x}$ *to* $\underline{y}$ *in* E **then**

      // to avoid introducing cycles

      $UE \leftarrow UE - \{\underline{x} - \underline{y}\}$

      $E \leftarrow E \cup \{\underline{x} \rightarrow \underline{y}\}$

  $newsize \leftarrow |UE|$

  if $size == newsize$ **then**

    $again \leftarrow False$

  else

    $size \leftarrow newsize$

return $G = (V, E, UE)$

Chapter 43

Turbo Codes

This chapter is based on Ref.[12].

In this chapter, vectors with n components will be indicated by an n superscript. For example, $a^n = (a_0, a_1, \ldots, a_{n-1})$.

Consider an n-letter message $u^n = (u_0, u_1, \ldots, u_{n-1})$, where for all i, $u_i \in \mathcal{A}$ is an element of an alphabet $\mathcal{A}$, and where for all i, the $\underline{u}_i$ are i.i.d.. Suppose u^n is encoded deterministically in two different ways, $e_1(u^n)$ and $e_2(u^n)$. After passing through the same memoryless channel, the variables u^n, e_1, e_2 become $\tilde{u}^n, \tilde{e}_1, \tilde{e}_2$, respectively. The letter u stands for unencoded, and e for encoded. Quantities with a tilde $\tilde{u}^n, \tilde{e}_1, \tilde{e}_2$ occur after channel passage and are visible (measurable). Quantities without a tilde u^n, e_1, e_2 are hidden (unmeasurable).

The situation just described can be represented by the bnet Fig.43.1, or by its abridged version Fig.43.2. But note that the abridged version does not show explicitly that the u_i are i.i.d. or that the channel is memoryless (i.e., that the u_i for all i pass independently through the channel).

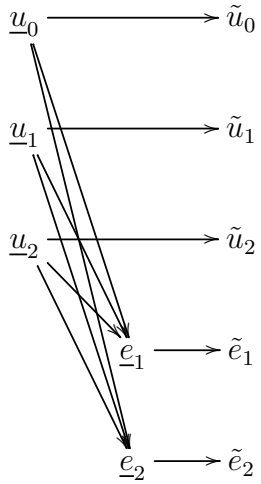


Figure 43.1: Turbo coding B net representing a message being encoded two different ways and then the original message and the 2 encodings pass through a memoryless channel.

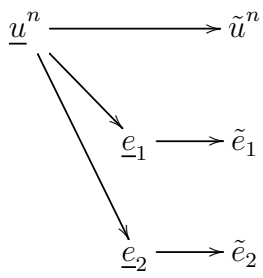


Figure 43.2: Abridged version of Fig.43.1.

Define

$$x = (u^n, e_1, e_2) \tag{43.1}$$

and

$$\tilde{x} = (\tilde{u}^n, \tilde{e}_1, \tilde{e}_2) \ . \tag{43.2}$$

Fig.43.1 implies that

$$P(x, \tilde{x}) = P(\tilde{u}^n | u^n) \left[\prod_{r=1,2} P(\tilde{e}_r | e_r) P(e_r | u^n) \right] P(u^n) \ . \tag{43.3}$$

Because the u^n are i.i.d.,

$$P(u^n) = \prod_i P(u_i) \ . \tag{43.4}$$

Because the channel is memoryless,

$$P(\tilde{u}^n | u^n) = \prod_i P(\tilde{u}_i | u_i) \ . \tag{43.5}$$

Because the encoding is deterministic, we must have for $r = 1, 2$

$$P(e_r | u^n) = \delta(e_r, e_r(u^n)) \ . \tag{43.6}$$

Define the belief functions

$$BEL_i = BEL_i(\underline{u}_i = a) = P(\underline{u}_i = a | \tilde{x}) \ . \tag{43.7}$$

The best estimate of u_j given all visible evidence $\tilde{x}$ is

$$\hat{u}_i = \operatorname*{argmax}_{u_i} BEL_i(u_i) \ . \tag{43.8}$$

Define the probability functions

$$\pi_i = \pi_i(u_i) = P(u_i) \ , \tag{43.9}$$

and the likelihood functions

$$\lambda_i = \lambda_i(u_i) = P(\tilde{u}_i|u_i) \ . \tag{43.10}$$

For $r = 1, 2$, define the Kernel functions

$$K_r = K_r(u^n) = P(\tilde{e}_r|e_r = e_r(u^n)) \ . \tag{43.11}$$

In this book, $\mathcal{N}(!a)$ denotes a normalization constant that does not depend on a. Define

$$\mathcal{N}_i = \mathcal{N}(!u_i) \ . \tag{43.12}$$

Claim 25

$$BEL_i = \mathcal{N}_i \lambda_i \pi_i \mathcal{T}_i^{K_1 K_2} [\prod_{j \neq i} \lambda_j \pi_j] \ , \tag{43.13}$$

where $\mathcal{T}_i^K(\cdot)$ with $K = K_1 K_2$ is an operator (transform) that acts on functions of u^n:

$$\mathcal{T}_i^K(\cdot) = \sum_{u^n} \delta(u_i, a) K(u^n)(\cdot) \ . \tag{43.14}$$

proof:

$$\begin{aligned}
&P(\underline{u}_i = a|\tilde{x}) = \\
&= \sum_x \delta(u_i, a) P(x|\tilde{x}) && (43.15)\\
&= \sum_x \delta(u_i, a) \frac{P(\tilde{x}|x)P(x)}{P(\tilde{x})} && (43.16)\\
&= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(\tilde{x}|x) P(x) && (43.17)\\
&= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(u^n) \left[\prod_{r=1,2} P(\tilde{e}_r|e_r) \delta(e_r, e_r(u^n)) \right] \prod_j P(\tilde{u}_j|u_j) && (43.18)\\
&= \mathcal{N}(!a) \lambda_i(a) \pi_i(a) R \ , && (43.19)
\end{aligned}$$

where

$$
\begin{aligned}
R &= \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} P(\tilde{e}_r | e_r(u^n)) \right] \prod_{j \neq i} P(\tilde{u}_j | u_j) P(u_j) && (43.20) \\
&= \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} K_r(u^n) \right] \prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j) && (43.21) \\
&= \mathcal{T}_i^{K_1 K_2} [\prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j)] \, . && (43.22)
\end{aligned}
$$

Hence

$$
BEL_i(a) = \mathcal{N}(!a) \lambda_i(a) \pi_i(a) \mathcal{T}_i^{K_1 K_2} [\prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j)] \, . \tag{43.23}
$$

QED

Decoding Algorithm

The Turbo algorithm for decoding the encode message is as follows. For $m = 0$, let

$$
\pi_j^{(0)}(u_j) = \frac{1}{n_{\underline{u}_j}} \, . \tag{43.24}
$$

Then for $m = 1, 2, \ldots,$ let

$$
\pi_i^{(m)} = \mathcal{N}_i \mathcal{T}_i^{K_{m\%2}} [\prod_{j \neq i} \lambda_j \pi_j^{(m-1)}] \, , \tag{43.25}
$$

where $m\%2 = 1$ if m is odd and $m\%2 = 2$ if m is even. Furthermore, for $m > 0$, let

$$
\begin{aligned}
BEL_i^{(m)} &= \mathcal{N}_i \lambda_i \pi_i^{(m-1)} \pi_i^{(m)} && (43.26) \\
&= \mathcal{N}_i \lambda_i \pi_i^{(m-1)} \mathcal{T}_i^{K_{m\%2}} [\prod_{j \neq i} \lambda_j \pi_j^{(m-1)}] \, . && (43.27)
\end{aligned}
$$

As $m \to \infty$, $BEL_i^{(m)}$ given by Eq.(43.27) is expected to converge to the the exact BEL_i given by Eq.(43.13).

Turbo decoding can be represented by the bnets Figs.43.3 and 43.4.

The node TPMs, printed in blue, for Fig.43.3, are given by:

$$
P(d_i^{(m)} = a \mid \tilde{u}^n, \tilde{e}_{m\%2}) = BEL_i^{(m)}(a) \, . \tag{43.28}
$$

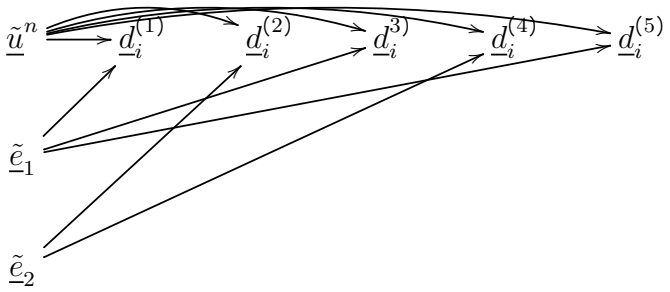


Figure 43.3: B net describing Turbo code generation of $BEL_i^{(m)}(a)$ for $m = 1, 2, \ldots$.

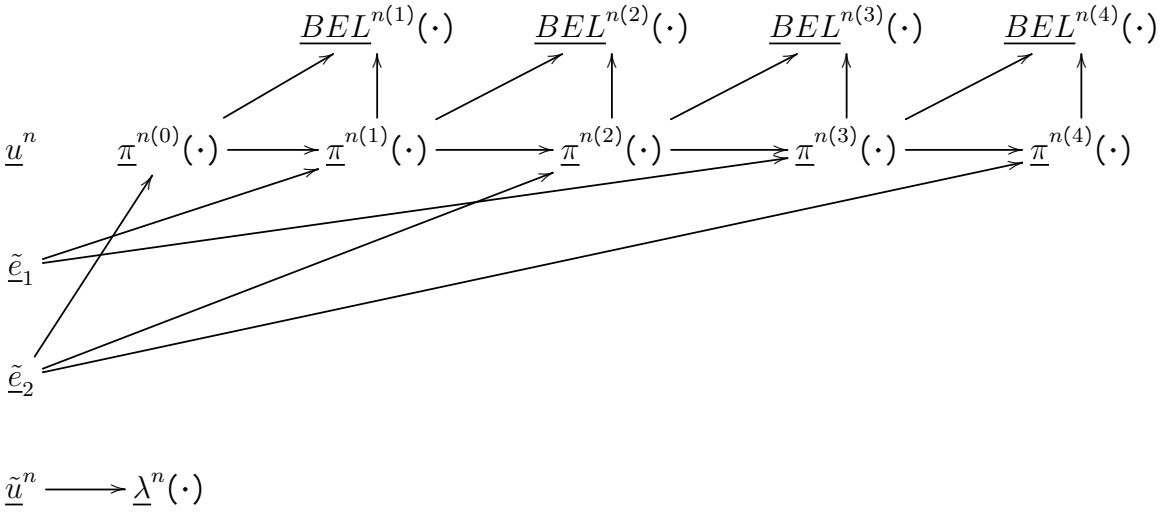


Figure 43.4: B net describing Turbo code generation of $BEL^{n(m)}(\cdot)$ and $\pi^{n(m)}(\cdot)$ for $m = 0, 1, 2 \ldots$. The following arrows were not drawn so as not to unduly clutter the diagram: Arrows pointing from node $\lambda^n(\cdot)$ to nodes $\pi^{n(m)}(\cdot)$ and $BEL^{n(m)}(\cdot)$ for $m = 0, 1, 2, \ldots$.

The TPMs, printed in blue, for Fig.43.4, are given by:

$$P((\lambda^n)'(\cdot)|\tilde{u}^n) = \delta((\lambda^n)'(\cdot), \lambda^n(\cdot)) \tag{43.29}$$

$$P(\pi^{n(m)}(\cdot)|\lambda^n(\cdot), \pi^{n(m-1)}(\cdot), \tilde{e}_{m\%2}) = \prod_i \prod_{u_i} \delta(\pi_i^{(m)}(u_i), \mathcal{N}_i \mathcal{T}_i^{K_{m\%2}}[\prod_{j \neq i} \lambda_j \pi_j^{(m-1)}]) \tag{43.30}$$

$$P(BEl^{n(m)}(\cdot)|\lambda^{n}(\cdot),\pi^{n(m)}(\cdot),\pi^{n(m-1)}(\cdot)) = \prod_{i}\prod_{u_i}\delta(BEL_i(u_i),\mathcal{N}_i\lambda_i\pi_i^{(m-1)}\pi_i^{(m)}) \tag{43.31}$$

Message Passing Interpretation of Decoding Algorithm

Ref.[12] shows that the Turbo code decoding algo can be interpreted as an application of Message Passing. We leave all talk of Message Passing to a separate chapter, Chapter 27.

Chapter 44

Variational Bayesian Approximation

For more info and references about this topic, see Ref.[62].

The Variational Bayesian approximation (VBA) is an analytic (as opposed to numerical) approximation to the probability distribution $P(h|\vec{x})$, where h are the hidden variables and $\vec{x}$ is the data.

More precisely, suppose $\underline{h} \in S_{\underline{h}}$ and $\underline{q} \in S_{\underline{h}}$ too. Suppose $\vec{\underline{x}} \in S_{\underline{x}}^{nsam}$ is a vector of $nsam$ samples and the samples $\underline{x}[\sigma] \in S_{\underline{x}}$ are i.i.d.. The VBA is simply an approximation $P_{\underline{q}|\vec{\underline{x}}}$ to $P_{\underline{h}|\vec{\underline{x}}}$:

$$P_{\underline{h}|\vec{\underline{x}}}(h|\vec{x}) \approx P_{\underline{q}|\vec{\underline{x}}}(h|\vec{x}) \tag{44.1}$$

obtained by minimizing the Kullback-Liebler divergence $D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \parallel P_{\underline{h}|\vec{\underline{x}}})$ over all $P_{\underline{q}|\vec{\underline{x}}}$. The minimization is usually subject to some constraints on the admissible forms of $P_{\underline{q}|\vec{\underline{x}}}$.

$D_{KL}(Q \parallel P) \neq D_{KL}(P \parallel Q)$; i.e., D_{KL} is not symmetric. So why do we use $D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \parallel P_{\underline{h}|\vec{\underline{x}}})$ instead of $D_{KL}(P_{\underline{h}|\vec{\underline{x}}} \parallel P_{\underline{q}|\vec{\underline{x}}})$? Because $D_{KL}(P_{\underline{h}|\vec{\underline{x}}} \parallel P_{\underline{q}|\vec{\underline{x}}})$ requires knowledge of $P_{\underline{h}|\vec{\underline{x}}}$, but calculating $P_{\underline{h}|\vec{\underline{x}}}$ is what we are trying to do in the first place.

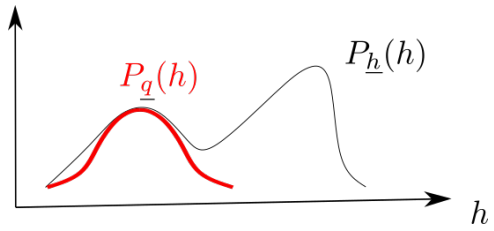


Figure 44.1: If $P_{\underline{q}}(h)$ is Gaussian shaped and $P_{\underline{h}}(h)$ has multiple bumps (modes) then $D_{KL}(P_{\underline{q}} \parallel P_{\underline{h}})$ is minimized when $P_{\underline{q}}$ fits one of the modes of $P_{\underline{h}}$. That is because $D_{KL}(P_{\underline{q}} \parallel P_{\underline{h}}) = \sum_h P_{\underline{q}}(h) \ln \frac{P_{\underline{q}}(h)}{P_{\underline{h}}(h)}$ is a weighted average with weights $P_{\underline{q}}$, so nothing going on outside the support of $P_{\underline{q}}$ influences much the final average.

See Fig.44.1 for some intuition on what minimizing $D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \parallel P_{\underline{h}|\vec{\underline{x}}})$ means.

Suppose $\underline{h} = (\underline{h}_0, \underline{h}_1, \ldots, \underline{h}_{nh-1})$ and $\underline{q} = (\underline{q}_0, \underline{q}_1, \ldots, \underline{q}_{nh-1})$ where $\underline{h}_i \in S_{\underline{h}_i}$ and $\underline{q}_i \in S_{\underline{h}_i}$ for

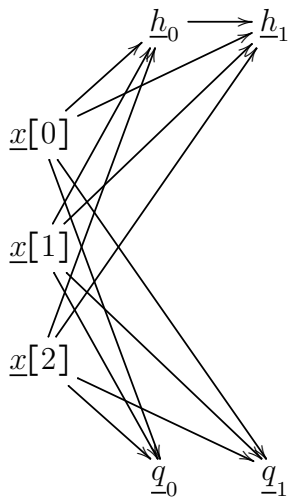


Figure 44.2: $\underline{q}$ and $\underline{h}$ have $nh = 2$ mirroring components and those of $\underline{q}$ are independent at fixed $\vec{\underline{x}}$.

all i. We say $\underline{q}$ and $\underline{h}$ have nh mirroring components and those of $\underline{q}$ are independent at fixed $\vec{\underline{x}}$ if

$$P_{\underline{q}|\vec{\underline{x}}}(h|\vec{x}) = \prod_i P_{\underline{q}_i|\vec{\underline{x}}}(h_i|\vec{x}) \ . \tag{44.2}$$

The bnet Fig.44.2 describes the scenario that we have in mind: The samples $\underline{x}[\sigma]$ are i.i.d.. Each component $\underline{h}_i$ of $\underline{h}$ has a mirroring component $\underline{q}_i$ in $\underline{q}$. The components of $\underline{h}$ are correlated whereas those of $\underline{q}$ are independent at fixed $\vec{\underline{x}}$.

Claim 26 *If $\underline{q}$ and $\underline{h}$ have nh mirroring components and those of $\underline{q}$ are independent at fixed $\vec{\underline{x}}$ and $D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \,||\, P_{\underline{h}|\vec{\underline{x}}})$ is minimum over all $P_{\underline{q}|\vec{\underline{x}}}$, then*

$$P_{\underline{q}_i|\vec{\underline{x}}}(q_i|\vec{x}) = \mathcal{N}(!q_i) e^{E_{(\underline{q}_j)_{j\neq i}}[\ln P_{\underline{h}|\vec{\underline{x}}}(\underline{h}=q|\vec{x})]} \tag{44.3}$$

$$= \mathcal{N}(!q_i) e^{E_{(\underline{q}_j)_{j\neq i}}[\ln P_{\underline{h},\vec{\underline{x}}}(\underline{h}=q,\vec{x})]} \tag{44.4}$$

for all i.

proof:

Since all quantities in Eq.(44.3) are conditioned on $\vec{x}$, let us omit all mention of $\vec{x}$ in this proof.

Let

$$\mathcal{L} = \mathcal{L}_0 + \mathcal{L}_1 \tag{44.5}$$

where

$$\begin{aligned}
\mathcal{L}_0 &= D_{KL}(P_{\underline{q}} \parallel P_{\underline{h}}) && (44.6)\\
&= \sum_h P_{\underline{q}}(h) \ln \frac{P_{\underline{q}}(h)}{P_{\underline{h}}(h)} && (44.7)\\
&= \sum_h P_{\underline{q}}(h) \ln P_{\underline{q}}(h) - \sum_h P_{\underline{q}}(h) \ln P_{\underline{h}}(h) && (44.8)\\
&= \sum_i \sum_{h_i} P_{\underline{q}_i}(h_i) \ln P_{\underline{q}_i}(h_i) - \sum_h P_{\underline{q}}(h) \ln P_{\underline{h}}(h) && (44.9)
\end{aligned}$$

and

$$\mathcal{L}_1 = \sum_i \lambda_i \left[\sum_{h_i} P_{\underline{q}_i}(h_i) - 1 \right] . \tag{44.10}$$

Then

$$\delta\mathcal{L} = \sum_i \sum_{h_i} \delta P_{\underline{q}_i}(h_i) \left[\ln P_{\underline{q}_i}(h_i) + 1 + \lambda_i - \frac{1}{nh} \sum_{(h_j)_{j\neq i}} \prod_{(h_j)_{j\neq i}} \{P_{\underline{q}_j}(h_j)\} \ln P_{\underline{h}}(h) \right] . \tag{44.11}$$

Hence,

$$P_{\underline{q}_i}(h_i) = \mathcal{N}(!h_i) e^{\sum_{(h_j)_{j\neq i}} \left\{ \prod_{(h_j)_{j\neq i}} P_{\underline{q}_j}(h_j) \right\} \ln P_{\underline{h}}(h)} . \tag{44.12}$$

QED

Note that Eq.(44.3) yields a system of nh nonlinear equations in nh unknowns $(P_{\underline{q}_i|\vec{\underline{x}}})_{i=0,1,\ldots,nh-1}$. This system is usually solved recursively.

Free Energy $\mathcal{F}(\vec{x})$

To simplify the notation below, let us introduce the following abbreviations:

$$P(h|\vec{x}) = P_{\underline{h}|\vec{\underline{x}}}(h|\vec{x}) \tag{44.13}$$

$$P(h,\vec{x}) = P_{\underline{h},\vec{\underline{x}}}(h,\vec{x}) \tag{44.14}$$

$$P(\vec{x}) = P_{\vec{\underline{x}}}(\vec{x}) \tag{44.15}$$

Note that

$$D_{KL}(P_{\underline{q}|\underline{\vec{x}}} \parallel P_{\underline{h}|\underline{\vec{x}}}) = \sum_h P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x})}{P(h|\vec{x})} \tag{44.16}$$

$$= \sum_h P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x})}{P(h,\vec{x})} + \ln P(\vec{x}) \tag{44.17}$$

$$= \mathcal{F}(\vec{x}) + \ln P(\vec{x}) \tag{44.18}$$

Hence, the **Free energy** $\mathcal{F}(\vec{x})$ is defined as

$$\mathcal{F}(\vec{x}) = \sum_h P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x})}{P(h,\vec{x})} \tag{44.19}$$

$$= E_{\underline{q}|\underline{\vec{x}}}\left[\ln \frac{P_{\underline{q}|\underline{\vec{x}}}(q|\vec{x})}{P_{\underline{h},\underline{\vec{x}}}(q,\vec{x})} \right] . \tag{44.20}$$

The name free energy is justified because

$$\mathcal{F}(\vec{x}) = \underbrace{-\sum_h P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x}) \ln P_{\underline{h},\underline{\vec{x}}}(h,\vec{x})}_{U,\text{ Internal Energy}} + \underbrace{\sum_h P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x}) \ln P_{\underline{q}|\underline{\vec{x}}}(h|\vec{x})}_{-S,\text{ minus Entropy}} . \tag{44.21}$$

It is also common to define a quantity called "ELBO" to be the negative of the free energy.

$$ELBO(\vec{x}) = -\mathcal{F}(\vec{x}) \tag{44.22}$$

ELBO stands for "Evidence Lower BOund". That name is justified because

$$\underbrace{\ln P_{\underline{\vec{x}}}(\vec{x})}_{\text{evidence}\leq 0} = \underbrace{D_{KL}(P_{\underline{q}|\underline{\vec{x}}} \parallel P_{\underline{h}|\underline{\vec{x}}})}_{\geq 0} - |ELBO(\vec{x})| . \tag{44.23}$$

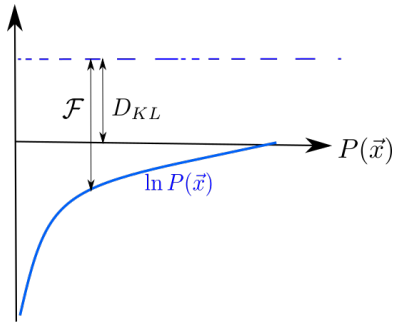


Figure 44.3: $D_{KL} + \ln \frac{1}{P(\vec{x})} = \mathcal{F}$.

Some properties of $\mathcal{F}$ are:

- **$\mathcal{F}$ is non-negative.**

$$\underbrace{D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \parallel P_{\underline{h}|\vec{\underline{x}}})}_{\geq 0} + \underbrace{\ln \frac{1}{P_{\vec{\underline{x}}}(\vec{x})]}}_{\geq 0} = \mathcal{F}(\vec{x}) \tag{44.24}$$

- **KL divergence is min iff $\mathcal{F}$ is min at fixed $P(\vec{x})$.**

  During a variation δ that holds $P(\vec{x})$ fixed, the KL divergence and $\mathcal{F}$ change by the same amount:

$$\delta D_{KL}(P_{\underline{q}|\vec{\underline{x}}} \parallel P_{\underline{h}|\vec{\underline{x}}}) = \delta \mathcal{F}(\vec{x}) \tag{44.25}$$

Chapter 45

Zero Information Transmission (Graphoid Axioms)

This chapter assumes that you have read Chapter 12 on d-separation.

The following quantities play a very prominent role in the d-separation Theorem that we enunciated in Chapter 12.

- the mutual information (MI)
  (aka information transmission) $H(\underline{a} : \underline{b})$
- the conditional mutual information (CMI)
  (aka conditional information transmission) $H(\underline{a} : \underline{b}|\underline{c})$

MI can be viewed as the special case of CMI, when the set of variables being conditioned on is empty. Particularly prominent in d-separation discussions are probability distributions for which CMI vanishes. The goal of this chapter is to study such probability distributions.

Recall that CMI is non-negative and symmetric in its first two variables (i.e., $H(\underline{a} : \underline{b}|\underline{c}) = H(\underline{b} : \underline{a}|\underline{c})$). Another very useful property of CMI is its chain rule (easy to prove from the definition of CMI):

$$H(\underline{y} : \underline{x}^n) = \sum_i H(\underline{y} : \underline{x}_i|\underline{x}_{<i}) \ , \tag{45.1}$$

where $\underline{x}^n = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{n-1})$ and $\underline{x}_{<i} = (\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{i-1})$.

A trivial but very useful consequence of the chain rule for CMI is:

$$\boxed{H(\underline{y} : \underline{x}^n) = 0 \implies H(\underline{y} : \underline{x}_i|\underline{x}_{<i}) = 0 \text{ for all } i} \ . \tag{45.2}$$

Consequences of Eq.45.2.

Table 45.1 gives a set of statements about CMI referred to as the Graphoid Axioms in chapter 1 of Ref.[19]. See Ref.[19] to learn the history of these axioms. The purpose of this section is to prove that the graphoid axioms are all a simple consequence of Eq.(45.2).

Symmetry	$\underline{a} \perp_P \underline{b} \implies \underline{b} \perp_P \underline{a}$ $H(\underline{a} : \underline{b}) = 0 \implies H(\underline{b} : \underline{a}) = 0$
Decomposition	$\underline{a} \perp_P \underline{b}, \underline{c} \implies \underline{a} \perp_P \underline{b}$ and $\underline{a} \perp_P \underline{c}$ $H(\underline{a} : \underline{b}, \underline{c}) = 0 \implies H(\underline{a} : \underline{b}) = 0$ and $H(\underline{a} : \underline{c}) = 0$
Weak Union	$\underline{a} \perp_P \underline{b}, \underline{c} \implies \underline{a} \perp_P \underline{b}\|\underline{c}$ and $\underline{a} \perp_P \underline{c}\|\underline{b}$ $H(\underline{a} : \underline{b}, \underline{c}) = 0 \implies H(\underline{a} : \underline{b}\|\underline{c}) = 0$ and $H(\underline{a} : \underline{c}\|\underline{b}) = 0$
Contraction	$\underline{a} \perp_P \underline{b}\|\underline{c}$ and $\underline{a} \perp_P \underline{c} \implies \underline{a} \perp_P \underline{b}, \underline{c}$ $H(\underline{a} : \underline{b}\|\underline{c}) = 0$ and $H(\underline{a} : \underline{c}) = 0 \implies H(\underline{a} : \underline{b}, \underline{c}) = 0$
Intersection	$\underline{a} \perp_P \underline{b}\|\underline{c}, \underline{d}$ and $\underline{a} \perp_P \underline{d}\|\underline{c}, \underline{b} \implies \underline{a} \perp_P \underline{b}, \underline{d}\|\underline{c}$ $H(\underline{a} : \underline{b}\|\underline{c}, \underline{d}) = 0$ and $H(\underline{a} : \underline{d}\|\underline{c}, \underline{b}) = 0 \implies H(\underline{a} : \underline{b}, \underline{d}\|\underline{c}) = 0$

Table 45.1: Graphoid Axioms

Claim 27 *Table 45.1 is true.*

proof:

- **Symmetry**

  Follows trivially from $H(\underline{a} : \underline{b}) = H(\underline{b} : \underline{a})$.

- **Decomposition**

  From the chain rule for CMI, we have

  $$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{b}|\underline{c}) + H(\underline{a} : \underline{c}) \ , \tag{45.3}$$

  and

  $$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{c}|\underline{b}) + H(\underline{a} : \underline{b}) \ . \tag{45.4}$$

  Hence,

  $$H(\underline{a} : \underline{b}, \underline{c}) = 0 \tag{45.5}$$

  implies

  $$H(\underline{a} : \underline{b}|\underline{c}) = H(\underline{a} : \underline{c}) = 0 \ , \tag{45.6}$$

  and

  $$H(\underline{a} : \underline{c}|\underline{b}) = H(\underline{a} : \underline{b}) = 0 \ . \tag{45.7}$$

- **Weak Union**

  Already proven in proof of Decomposition.

- **Contraction**

  From chain rule for CMI, we have

  $$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{b}|\underline{c}) + H(\underline{a} : \underline{c}) \ . \tag{45.8}$$

- **Intersection**

  From the chain rule for CMI, we have

$$H(\underline{a} : \underline{b}, \underline{d}|\underline{c}) = H(\underline{a} : \underline{b}|\underline{d}, \underline{c}) + H(\underline{a} : \underline{d}|\underline{c}) \ , \tag{45.9}$$

  and

$$H(\underline{a} : \underline{b}, \underline{d}|\underline{c}) = H(\underline{a} : \underline{d}|\underline{b}, \underline{c}) + H(\underline{a} : \underline{b}|\underline{c}) \ . \tag{45.10}$$

  Thus,

$$H(\underline{a} : \underline{b}, \underline{d}|\underline{c}) = 0 \tag{45.11}$$

  implies

$$H(\underline{a} : \underline{b}|\underline{d}, \underline{c}) = H(\underline{a} : \underline{d}|\underline{c}) = 0 \ , \tag{45.12}$$

  and

$$H(\underline{a} : \underline{d}|\underline{b}, \underline{c}) = H(\underline{a} : \underline{b}|\underline{c}) = 0 \ . \tag{45.13}$$

.

QED

Bibliography

[1] Dan Bendel. Metropolis-Hastings: A comprehensive overview and proof. https://similarweb.engineering/mcmc/.

[2] Alexandra M Carvalho. Scoring functions for learning Bayesian networks. http://www.lx.it.pt/~asmc/pub/talks/09-TA/ta_pres.pdf.

[3] Charles Fox, Neil Girdhar, and Kevin Gurney. A causal Bayesian network view of reinforcement learning. https://www.aaai.org/Papers/FLAIRS/2008/FLAIRS08-030.pdf".

[4] Bruno Gonçalves. Model testing and causal search. blog post https://medium.com/data-for-science/causal-inference-part-vii-model-testing-and-causal-search-536b796f

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, David Warde-Farley Bing Xu, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. https://arxiv.org/abs/1406.2661.

[6] Christina Heinze-Deml. Causality, spring semester 2019 at ETH Zurich. https://stat.ethz.ch/lectures/ss19/causality.php#course_materials.

[7] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, 15(3):225–263, 1996. http://www.ar-tiste.com/Huang-Darwiche1996.pdf.

[8] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988. http://www.eecis.udel.edu/~shatkay/Course/papers/Lauritzen1988.pdf.

[9] Sergey Levine. Course CS 285 at UC Berkeley, Deep reinforcement learning. http://rail.eecs.berkeley.edu/deeprlcourse/.

[10] Dimitris Margaritis. Learning Bayesian network model structure from data (thesis, 2003, Carnegie Mellon Univ). https://apps.dtic.mil/sti/citations/ADA461103.

[11] Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. In *BMC bioinformatics*, volume 7, page S7. Springer, 2006. https://link.springer.com/article/10.1186/1471-2105-7-S1-S7.

[12] Robert J. McEliece, David J. C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearls belief propagation algorithm. http://authors.library.caltech.edu/6938/1/MCEieeejstc98.pdf.

[13] Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, 2004.

[14] Andrew Ng. Lecture at deeplearning.ai on recurrent neural networks. http://www.ar-tiste.com/ng-lec-rnn.pdf.

[15] Gregory Nuel. Tutorial on exact belief propagation in Bayesian networks: from messages to algorithms. https://arxiv.org/abs/1201.4724.

[16] Judea Pearl. Mediating instrumental variables. https://ftp.cs.ucla.edu/pub/stat_ser/r210.pdf.

[17] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. https://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf, 1982.

[18] Judea Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, 1988.

[19] Judea Pearl. *Causality: Models, Reasoning, and Inference, Second Edition*. Cambridge University Press, 2013.

[20] Judea Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, pages 1–41, 2019. https://ftp.cs.ucla.edu/pub/stat_ser/r485.pdf.

[21] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

[22] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[23] ReliaSoft. System analysis reference. http://reliawiki.org/index.php/System_Analysis_Reference.

[24] Marco Scutari. bnlearn. https://www.bnlearn.com/.

[25] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253, 2019. https://arxiv.org/abs/1805.11908.

[26] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[27] Masayoshi Takahashi. Statistical inference in missing data by MCMC and non-MCMC multiple imputation algorithms: Assessing the effects of between-imputation iterations. *Data Science Journal*, 16, 2017. https://datascience.codata.org/articles/10.5334/dsj-2017-037/.

[28] theinvestorsbook.com. Pert analysis. https://theinvestorsbook.com/pert-analysis.html.

[29] Robert R. Tucci. Bell's inequalities for Bayesian statisticians. blog post in blog Quantum Bayesian Networks, https://qbnets.wordpress.com/2008/09/19/bells-inequaties-for-bayesian-statistician/.

[30] Robert R. Tucci. Quantum Fog. https://github.com/artiste-qb-net/quantum-fog.

[31] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl. Fault tree handbook nureg-0492. https://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/.

[32] Wikipedia. Belief propagation. https://en.wikipedia.org/wiki/Belief_propagation.

[33] Wikipedia. Beta function. https://en.wikipedia.org/wiki/Beta_function.

[34] Wikipedia. Binary decision diagram. https://en.wikipedia.org/wiki/Binary_decision_diagram.

[35] Wikipedia. Boolean algebra. https://en.wikipedia.org/wiki/Boolean_algebra.

[36] Wikipedia. Categorical distribution. https://en.wikipedia.org/wiki/Categorical_distribution.

[37] Wikipedia. Chow-Liu tree. https://en.wikipedia.org/wiki/Chow%E2%80%93Liu_tree.

[38] Wikipedia. Data processing inequality. https://en.wikipedia.org/wiki/Data_processing_inequality.

[39] Wikipedia. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution.

[40] Wikipedia. Expectation maximization. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

[41] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function.

[42] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit.

[43] Wikipedia. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling.

[44] Wikipedia. Hidden Markov model. https://en.wikipedia.org/wiki/Hidden_Markov_model.

[45] Wikipedia. Importance sampling. https://en.wikipedia.org/wiki/Importance_sampling.

[46] Wikipedia. Inverse transform sampling. https://en.wikipedia.org/wiki/Inverse_transform_sampling.

[47] Wikipedia. Junction tree algorithm. https://en.wikipedia.org/wiki/Junction_tree_algorithm.

[48] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.

[49] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter.

[50] Wikipedia. Long short term memory. https://en.wikipedia.org/wiki/Long_short-term_memory.

[51] Wikipedia. Markov blanket. https://en.wikipedia.org/wiki/Markov_blanket.

[52] Wikipedia. Metropolis-Hastings method. https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm.

[53] Wikipedia. Minimum spanning tree. https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[54] Wikipedia. Monte Carlo methods. https://en.wikipedia.org/wiki/Category:Monte_Carlo_methods.

[55] Wikipedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.

[56] Wikipedia. Multinomial theorem. https://en.wikipedia.org/wiki/Multinomial_theorem.

[57] Wikipedia. Multivariate normal distribution. https://en.wikipedia.org/wiki/Multivariate_normal_distribution.

[58] Wikipedia. Non-negative matrix factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.

[59] Wikipedia. Program evaluation and review technique. https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique.

[60] Wikipedia. Rejection sampling. https://en.wikipedia.org/wiki/Rejection_sampling.

[61] Wikipedia. Simpson's paradox. https://en.wikipedia.org/wiki/Simpson's_paradox.

[62] Wikipedia. Variational Bayesian methods. https://en.wikipedia.org/wiki/Variational_Bayesian_methods.

[63] Hao Wu and Zhaohui Steve Qin. course notes, BIOS731: Advanced statistical computing, 2016 Emory Univ. http://web1.sph.emory.edu/users/hwu30/teaching/statcomp/statcomp.html.