

Bayesuvius,
a small visual dictionary of Bayesian Networks

Robert R. Tucci
www.ar-tiste.xyz

May 5, 2021



Figure 1: View of Mount Vesuvius from Pompeii



Figure 2: Mount Vesuvius and Bay of Naples

Contents

Foreword	10
Navigating the ocean of Judea Pearl's Books	11
Notational Conventions and Preliminaries	12
0.1 Some abbreviations frequently used throughout this book	12
0.2 $\mathcal{N}(!a)$	12
0.3 One hot	12
0.4 Special sets	12
0.5 Kronecker delta function	13
0.6 Dirac delta function	13
0.7 Indicator function (aka Truth function)	13
0.8 Majority function	13
0.9 Underlined letters indicate random variables	13
0.10 Probability distributions	14
0.11 Discretization of continuous probability distributions	14
0.12 Samples, i.i.d. variables	14
0.13 Normal Distribution	15
0.14 Uniform Distribution	15
0.15 Sigmoid and logit functions	16
0.16 Expected Value and Variance	16
0.17 Conditional Expected Value	16
0.18 Law of Total Variance	17
0.19 Notation for covariances	18
0.20 Conditional Covariance	18
0.21 Linear regression, Ordinary Least Squares (OLS)	19
0.21.1 LR, assuming x_σ are non-random	19
0.21.2 LR, assuming x_σ are random and i.i.d.	22
0.22 Entropy, Kullback-Liebler divergence	24
0.23 Definition of various entropies used in Shannon Information Theory	24
0.24 Pearson Chi-Squared Test	25
0.25 Demystifying Population and Sample Variances	26
0.26 Error Bars	28

0.27 Short Summary of Boolean Algebra	30
Definition of a Bayesian Network	32
1 AdaBoost	35
1.1 AdaBoost for general ensemble of w-classifiers	35
1.2 AdaBoost for ensemble of tree stumps	39
2 ARACNE structure learning	41
3 Backdoor Adjustment	43
3.1 Examples	44
4 Back Propagation (Automatic Differentiation)	47
4.1 General Theory	47
4.1.1 Jacobians	47
4.1.2 bnets for function composition, forward propagation and back propagation	48
4.2 Application to Neural Networks	49
4.2.1 Absorbing b_i^λ into w_{ij}	49
4.2.2 bnets for function composition, forward propagation and back propagation for NN	51
4.3 General bnets instead of Markov chains induced by layered structure of NNs	54
5 Basic Curve Fitting Using Gradient Descent	55
6 Bell and Clauser-Horne Inequalities in Quantum Mechanics	57
7 Berkson's Paradox	58
8 Binary Decision Diagrams	60
9 Chow-Liu Trees and Tree Augmented Naive Bayes (TAN)	64
9.1 Chow-Liu Trees	64
9.2 Tree Augmented Naive Bayes (TAN)	68
10 Counterfactual Reasoning	70
10.1 The 3 Rungs of Causal AI	70
10.2 Two kinds of intervention operators	70
10.3 Do operator for DEN diagrams	72
10.4 Mediation Analysis	74
11 Cross-Validation	78

12 Decision Trees	81
12.1 Transforming a dtree into a bnet	82
12.2 Structure Learning for Dtrees	84
12.2.1 Information Gain, Gini	84
12.3 Information Gain Ratio	88
12.3.1 Pseudo-code	88
13 Difference-in-Differences	90
13.1 John Snow, DID and a cholera transmission pathway	90
13.2 PO analysis	92
13.3 Linear Regression	95
14 Digital Circuits	97
14.1 Mapping any dcircuit to a bnet	98
14.1.1 Option A of Fig.14.2	98
14.1.2 Option B of Fig.14.2	98
15 Do-Calculus	99
15.1 Parent Adjustment	102
15.2 3 Rules of do-calculus	103
15.3 Backdoor Adjustment	104
15.4 Front Door Adjustment	105
16 D-Separation	109
17 Dynamic Bayesian Networks	112
18 Expectation Maximization	114
18.1 The EM algorithm:	115
18.1.1 Motivation	116
18.2 Minorize-Maximize (MM) algorithms	117
18.3 Examples	118
18.3.1 Gaussian mixture	118
18.3.2 Blood Genotypes and Phenotypes	119
18.3.3 Missing Data/Imputation	121
19 Front-door Adjustment	122
19.1 Examples	123
20 Gaussian Nodes with Linear Dependence on Parents	124
21 Generative Adversarial Networks (GANs)	127

22 Goodness of Causal Fit	132
22.1 Abstract	132
22.2 Introduction	132
22.3 Goodness of Fit	133
22.4 GCF example 1	134
22.5 GCF example 2	136
22.6 GCF in general	137
23 Hidden Markov Model	138
24 Influence Diagrams & Utility Nodes	142
25 Instrumental Inequality and beyond	144
25.1 I-inequality	144
25.1.1 I-inequality for binary z,d,y	146
25.2 Bounds on Effect of IV on treatment outcome y	147
26 Instrumental Variables	150
26.1 δ with unmeasured confounder	150
26.2 δ (with unmeasured confounder) can be inferred via IV	151
26.3 More general bnets with IVs	153
26.4 Instrumental Inequality	153
27 Jackknife Resampling	154
27.1 Case $A = A^n(\vec{x}) = \frac{1}{n} \sum_{\sigma} x^{\sigma}$	156
28 Junction Tree Algorithm	158
29 Kalman Filter	159
29.1 Problem	160
29.2 Solution	160
30 Linear and Logistic Regression	162
30.1 Generalization to x with multiple components (features)	164
30.2 Alternative $V(b, m)$ for logistic regression	164
31 Linear Deterministic Bnets with External Noise	166
31.1 Example of LDEN diagram	166
31.2 Fully Connected LDEN diagrams	167
31.2.1 Fully connected LDEN diagram with $nx = 2$	168
31.2.2 Fully connected LDEN diagram with $nx = 3$	168
31.2.3 Fully connected LDEN diagram with arbitrary nx	169
31.3 Non-linear DEN diagrams	171

32 Markov Blankets	172
33 Markov Chain Monte Carlo (MCMC)	174
33.1 Inverse Cumulative Sampling	174
33.2 Rejection Sampling	176
33.3 Metropolis-Hastings Sampling	177
33.4 Gibbs Sampling	180
33.5 Importance Sampling	181
34 Markov Chains	183
35 Message Passing (Belief Propagation)	184
35.1 Distributed Soldier Counting	184
35.2 Spring Systems	186
35.3 BP for Markov Chains	186
35.4 BP Algorithm for Polytrees	193
35.4.1 How BP algo for polytrees reduces to the BP algo for Markov chains	197
35.5 Derivation of BP Algorithm for Polytrees	198
35.6 Example of BP algo for a Tree	201
35.7 Bipartite bnets	205
35.8 BP for bipartite bnets (BP-BB)	208
35.8.1 BP-BB and general BP agree on Markov chains	209
35.8.2 BP-BB and general BP agree on tree bnets.	211
35.9 BP-BB and sum-product decomposition	213
36 Missing Data, Imputation	214
36.1 Imputation via EM	215
36.2 Imputation via MCMC	218
36.3 Multiple Imputations	219
37 Monty Hall Problem	220
38 Naive Bayes	222
39 Neural Networks	223
39.1 Activation Functions $\mathcal{A}_i^\lambda : \mathbb{R} \rightarrow \mathbb{R}$	224
39.2 Weight optimization via supervised training and gradient descent . .	226
39.3 Non-dense layers	228
39.4 Autoencoder NN	229
40 Noisy-OR gate	230
40.1 3 ways to interpret the parameters π_i	231

41 Non-negative Matrix Factorization	234
41.1 Bnet interpretation	234
41.2 Simplest recursive algorithm	235
42 Observationally Equivalent DAGs	236
42.1 Examples	236
43 Potential Outcomes	239
43.1 G and G_{den} , bnets, the starting point bnets	240
43.2 G_{do+} bnet	242
43.3 G_{im+} bnet	243
43.4 G_{im+} bnet with nodes $y^\sigma(0), y^\sigma(1)$ added to it.	244
43.5 Conditional Independence Assumption	246
43.6 $\mathcal{Y}_{ \tilde{d},x}$ and G_{do}	247
43.7 Translation Dictionary	248
43.8 $\mathcal{Y}_{d \tilde{d}}$ differences (aka treatment effects)	248
43.9 Zero ACE, $\mathcal{Y}_{1 0} = \mathcal{Y}_1$	250
43.10(SDO, ATE) space	251
43.11Matching Strata	253
43.11.1 Exact strata-match	253
Example, calculation of estimators for a treatment	255
43.11.2 Approximate strata-match	257
43.11.3 Positivity	257
43.12Propensity Score	258
43.13Multi-time PO bnets (Panel Data)	260
44 Program evaluation and review technique (PERT)	263
44.1 Example	265
45 Random Forest and Bagging	269
45.1 Bagging (with fully-featured bags)	269
45.2 Bagging (with randomly-shortened bags)	271
46 Recurrent Neural Networks	272
46.1 Language Sequence Modeling	275
46.2 Other types of RNN	275
46.2.1 Long Short Term Memory (LSTM) unit (1997)	277
46.2.2 Gated Recurrence Unit (GRU) (2014)	279
47 Regression Discontinuity Design	281
47.1 PO analysis	281
47.2 Linear Regression	282

48 Reinforcement Learning (RL)	284
48.1 Exact RL bnet	287
48.2 Actor-Critic RL bnet	289
48.3 Q function learning RL bnet	291
49 Reliability Box Diagrams and Fault Tree Diagrams	293
49.1 Minimal Cut Sets	299
50 Restricted Boltzmann Machines	301
51 ROC curves	303
51.1 Terminology Table Adapted from Wikipedia Ref.[86]	306
52 Scoring the Nodes of a Learned Bnet	308
52.1 Probability Distributions and Special Functions	309
52.2 Single node with no parents	311
52.3 Multiple nodes with any number of parents	313
52.4 Bayesian Scores	315
52.5 Information Theoretic scores	315
53 Simpson's Paradox	317
53.1 Pearl Causality	319
53.2 Numerical Example	320
54 Structure and Parameter Learning for Bnets	321
54.1 Overview	321
54.2 Score based SL algorithms	323
54.3 Constraint based SL algorithms	324
54.4 Pseudo-code for some bnet learning algorithms	325
55 Support Vector Machines And Kernel Method	327
55.1 Learning Algorithm for SVM Classifier	328
55.2 Linear (dot-product) Kernel	329
55.3 Alternatives to Linear Kernel	332
55.4 Random Forest and Kernel Method	333
56 Synthetic Controls	334
56.1 A bnet G_t with weighted treatment outcomes	336
56.2 PO analysis	337
57 Transformer Networks	339
58 Transportability: COMING SOON	342

59 Turbo Codes	344
59.1 Decoding Algorithm	347
59.2 Message Passing Interpretation of Decoding Algorithm	349
60 Uplift Modelling	350
60.1 UP types	350
60.2 Some Relevant Technical Facts from Chapter 43	352
60.3 UP Analysis	352
60.4 UP Decision Trees	354
60.4.1 Appendix, connection between Δ_c and $\Delta_{c j}$	359
61 Variational Bayesian Approximation	360
61.1 Free Energy $\mathcal{F}(\vec{x})$	362
62 Zero Information Transmission (Graphoid Axioms)	365
62.1 Consequences of Eq.(62.2)	365
Bibliography	368

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 Bnet. 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 Bnets (Linear Regression and GAN). I will add more chapters periodically. Remember, this is a moon-lighting 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

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.[26] 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.[27].
2. In 2016, he released together with Glymour and Jewell, a less advanced “primer” on causality, Ref.[30].
3. In 2018, he released together with Mackenzie his lovely “The Book of Why”, Ref.[31].

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.

A micro poem written by me to celebrate Judea Pearl and his work:

I, Robot

Let other robots `talk()`,
while I,
`talk()`, `do()` and `imagine()`.

Notational Conventions and Preliminaries

0.1 Some abbreviations frequently used throughout this book

- bnet= Bnet= Bayesian Network
- CPT = Conditional Probabilities Table, same as TPM
- DAG = Directed Acyclic Graph
- i.i.d.= independent identically distributed.
- RCT= Randomized Controlled Trial, aka A/B testing.
- TPM= Transition Probability Matrix, same as CPT

0.2 $\mathcal{N}(!a)$

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

0.3 One hot

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 $x^n = (x_0, x_1, \dots, x_{n-1})$ and $x_i = \delta(i, 0)$ then x^n is one hot.

0.4 Special sets

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).

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

0.5 Kronecker delta function

For x, y in discrete set S ,

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

0.6 Dirac delta function

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

$$\int_{-\infty}^{+\infty} dx \delta(x - y) f(x) = f(y) \quad (2)$$

0.7 Indicator function (aka Truth function)

$$\mathbb{1}(\mathcal{S}) = \begin{cases} 1 & \text{if } \mathcal{S} \text{ is true} \\ 0 & \text{if } \mathcal{S} \text{ is false} \end{cases} \quad (3)$$

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

0.8 Majority function

The **majority function** is defined as follows.

$$\text{majority}(L) = \begin{array}{l} \text{most common element of list } L \\ (\text{ties resolved by chance}) \end{array} \quad (4)$$

Note that the majority function acts on lists, not sets. By definition, all elements of a set appear only once in the set. $\text{majority}(L)$ is usually used when the elements of L are categorical (i.e., not real numbers). When they are real numbers, it makes more sense to use, instead of $\text{majority}(L)$, a simple average of the elements of L .

0.9 Underlined letters indicate random variables

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.

0.10 Probability distributions

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

$$\sum_{x \in S_{\underline{x}}} P_{\underline{x}}(x) = 1 \quad (5)$$

$$P_{\underline{x}, \underline{y}}(x, y) = P(\underline{x} = x, \underline{y} = y) = P(x, y) \quad (6)$$

$$P_{\underline{x}|\underline{y}}(x|y) = P(\underline{x} = x|\underline{y} = y) = P(x|y) = \frac{P(x, y)}{P(y)} \quad (7)$$

0.11 Discretization of continuous probability distributions

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 $f : [a, b] \rightarrow \mathbb{R}$. 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

$$\frac{1}{(b-a)} \int_{[a,b]} dx f(x) \rightarrow \frac{\Delta x}{(b-a)} \sum_{x \in S_{\underline{x}}} f(x) = \frac{1}{n_{\underline{x}}} \sum_{x \in S_{\underline{x}}} f(x) . \quad (8)$$

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

$$\int_{[a,b]} dx \delta(x - y) f(x) = f(y) \rightarrow \sum_{x \in S_{\underline{x}}} \delta(x, y) f(x) = f(y) . \quad (9)$$

0.12 Samples, i.i.d. variables

$$\vec{x} = (x[0], x[1], x[2] \dots, x[n_{sam}(\vec{x}) - 1]) = x[:] \quad (10)$$

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

$$\underline{x}[\sigma] \sim P_{\underline{x}} \text{ (i.e. } P_{\underline{x}[\sigma]} = P_{\underline{x}}) \quad (11)$$

$$P(\underline{x} = x) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x[\sigma] = x) \quad (12)$$

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

$$\sum_x P(\underline{x} = x) f(x) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} f(x[\sigma]) \quad (13)$$

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.

$$P(\vec{x}) = \prod_{\sigma} P(x[\sigma]) \quad (14)$$

$$\sum_{\vec{x}} = \prod_{\sigma} \sum_{x[\sigma]} \quad (15)$$

$$\partial_{\vec{x}} = [\partial_{x[0]}, \partial_{x[1]}, \partial_{x[2]}, \dots, \partial_{x[nsam(\vec{x})-1]}] \quad (16)$$

$$P(\vec{x}) \approx [\prod_x P(x)^{P(x)}]^{nsam(\vec{x})} \quad (17)$$

$$= e^{nsam(\vec{x}) \sum_x P(x) \ln P(x)} \quad (18)$$

$$= e^{-nsam(\vec{x}) H(P_{\underline{x}})} \quad (19)$$

0.13 Normal Distribution

For $x, \mu, \sigma \in \mathbb{R}$, $\sigma > 0$

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2} \quad (20)$$

0.14 Uniform Distribution

For $a < b$, $x \in [a, b]$

$$\mathcal{U}(x; a, b) = \frac{1}{b-a} \quad (21)$$

0.15 Sigmoid and logit functions

The sigmoid function $\text{sig} : \mathbb{R} \rightarrow [0, 1]$ is defined by

$$\text{smoid}(x) = \frac{1}{1 + e^{-x}} \quad (22)$$

$\text{smoid}()$ is monotonically increasing with $\text{smoid}(-\infty) = 0$ and $\text{smoid}(+\infty) = 1$.

The logit or log-odds function $\text{logit} : [0, 1] \rightarrow \mathbb{R}$ is defined by

$$\text{logit}(p) = \ln \frac{p}{1 - p} \quad (23)$$

$\text{logit}()$ is the inverse of $\text{smoid}()$:

$$\text{logit}[\text{smoid}(x)] = x \quad (24)$$

0.16 Expected Value and Variance

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

$$E_{\underline{x}}[f(\underline{x})] = E_{x \sim P(x)}[f(x)] = \sum_x P(x)f(x) \quad (25)$$

$$Var_{\underline{x}}[f(\underline{x})] = E_{\underline{x}}[(f(\underline{x}) - E_{\underline{x}}[f(\underline{x})])^2] \quad (26)$$

$$= E_{\underline{x}}[f(\underline{x})^2] - (E_{\underline{x}}[f(\underline{x})])^2 \quad (27)$$

$$E[\underline{x}] = E_{\underline{x}}[\underline{x}] \quad (28)$$

$$Var[\underline{x}] = Var_{\underline{x}}[\underline{x}] \quad (29)$$

0.17 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 $f : S_{\underline{x}} \times S_{\underline{y}} \rightarrow \mathbb{R}$, define

$$E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})] = \sum_x P(x|\underline{y})f(x, \underline{y}), \quad (30)$$

$$E_{\underline{x}|\underline{y}=y}[f(\underline{x}, y)] = E_{\underline{x}|y}[f(\underline{x}, y)] = \sum_x P(x|y)f(x, y). \quad (31)$$

Note that

$$E_{\underline{y}}[E_{\underline{x}|\underline{y}}[f(\underline{x}, \underline{y})]] = \sum_{x,y} P(x|y)P(y)f(x, y) \quad (32)$$

$$= \sum_{x,y} P(x, y)f(x, y) \quad (33)$$

$$= E_{\underline{x}, \underline{y}}[f(\underline{x}, \underline{y})]. \quad (34)$$

0.18 Law of Total Variance

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

$$Var_{\underline{x}, \underline{y}}(f) = E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(f)] + Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[f]). \quad (35)$$

In particular,

$$Var_{\underline{x}}(x) = E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(x)] + Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[x]). \quad (36)$$

proof:

Let

$$A = \sum_y P(y) \left(\sum_x P(x|y)f \right)^2. \quad (37)$$

Then

$$Var_{\underline{x}, \underline{y}}(f) = \sum_{x,y} P(x, y)f^2 - \left(\sum_{x,y} P(x, y)f \right)^2 \quad (38)$$

$$= \begin{cases} \sum_{x,y} P(x, y)f^2 - A \\ + \left(A - \left(\sum_{x,y} P(x, y)f \right)^2 \right) \end{cases} \quad (39)$$

$$E_{\underline{y}}[Var_{\underline{x}|\underline{y}}(f)] = \sum_y P(y) \left(\sum_x P(x|y)f^2 - \left(\sum_x P(x|y)f \right)^2 \right) \quad (40)$$

$$= \sum_{x,y} P(x, y)f^2 - A \quad (41)$$

$$Var_{\underline{y}}(E_{\underline{x}|\underline{y}}[f]) = \sum_y P(y) \left(\sum_x P(x|y)f \right)^2 - \left(\sum_y P(y) \sum_x P(x|y)f \right)^2 \quad (42)$$

$$= A - \left(\sum_{x,y} P(x, y)f \right)^2 \quad (43)$$

QED

0.19 Notation for covariances

Consider two random variables $\underline{x}, \underline{y}$.

- Mean value of \underline{x}

$$\langle \underline{x} \rangle = E_{\underline{x}}[\underline{x}] \quad (44)$$

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

$$\Delta \underline{x} = \underline{x} - \langle \underline{x} \rangle \quad (45)$$

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

$$Cov(\underline{x}, \underline{y}) = \langle \underline{x}, \underline{y} \rangle = \langle \Delta \underline{x} \Delta \underline{y} \rangle = \langle \underline{x} \underline{y} \rangle - \langle \underline{x} \rangle \langle \underline{y} \rangle \quad (46)$$

$\langle \underline{x}, \underline{y} \rangle$ is symmetric (i.e., $\langle \underline{x}, \underline{y} \rangle = \langle \underline{y}, \underline{x} \rangle$) and bilinear (i.e., $\langle \sum_i \alpha_i \underline{x}_i, \underline{y} \rangle = \sum_i \alpha_i \langle \underline{x}_i, \underline{y} \rangle$, where $\alpha_i \in \mathbb{R}$ are non-random scalars and $\underline{x}_i, \underline{y} \in \mathbb{R}$ are real-valued random variables.)

- Variance of \underline{x}

$$Var(\underline{x}) = \langle \underline{x}, \underline{x} \rangle \quad (47)$$

- Standard deviation or \underline{x}

$$\sigma_{\underline{x}} = \sqrt{\langle \underline{x}, \underline{x} \rangle} \quad (48)$$

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

$$\rho_{\underline{x}, \underline{y}} = \frac{\langle \underline{x}, \underline{y} \rangle}{\sqrt{\langle \underline{x}, \underline{x} \rangle \langle \underline{y}, \underline{y} \rangle}} \quad (49)$$

0.20 Conditional Covariance

Let $\underline{x}, \underline{y}, \underline{a}$ be random variables. The covariance $Cov(\underline{x}, \underline{y}|\underline{a})$ of \underline{x} and \underline{y} given \underline{a} , is defined the same way as $Cov(\underline{x}, \underline{y})$, except that all expected values are conditioned on \underline{a} .

$$Cov(\underline{x}, \underline{y}|\underline{a}) = \langle \underline{x}, \underline{y} \rangle_{|\underline{a}} = \left\langle (\underline{x} - \langle \underline{x} \rangle_{|\underline{a}})(\underline{y} - \langle \underline{y} \rangle_{|\underline{a}}) \right\rangle_{|\underline{a}} \quad (50)$$

where

$$\langle \underline{x} \rangle_{|\underline{a}} = E_{\underline{x}|\underline{a}}[\underline{x}] . \quad (51)$$

0.21 Linear regression, Ordinary Least Squares (OLS)

Wikipedia articles

1. Linear Regression (LR)

- linear regression, Ref.[72]
- simple linear regression, Ref.[88]
- errors in variable, Ref.[56]

2. Least squares (LS)

- least squares, Ref.[71]
- ordinary least squares (OLS), Ref.[83]

In LR, the **dependent variables** y equal a linear combination of some **independent variables** x plus some external noise variables ϵ called the **residuals**.

Below, we consider two types of LR:

1. LR in which the independent variables are non-random.
2. LR in which the independent variables are random and i.i.d.

Once one assumes that certain variables are random, a “model” (i.e., a bnet) for the random variables must be specified.

For LR of type 2, there is randomness in y coming from the randomness in x and in the residuals. For LR of type 1, there is randomness in y too, but it comes from the residuals only.

OLS provides a cost function which when minimized, yields LR. The term OLS is often used to refer to LR of type 1.

0.21.1 LR, assuming x_σ are non-random

Let

$\sigma \in \{0, 1, 2, \dots, nsam - 1\}$: sample index

$y_\sigma \in \mathbb{R}$: dependent variables

$x_{\sigma j} \in \mathbb{R}$: independent variables

$\epsilon_\sigma \in \mathbb{R}$: residuals

$\beta_0, \beta_j \in \mathbb{R}$: regression coefficients

$$y_\sigma = \beta_0 + \sum_{j=1}^n x_{\sigma j} \beta_j + \epsilon_\sigma \quad (52)$$

If we define

$$x_{\sigma 0} = 1 \quad (53)$$

for all σ , then

$$y_\sigma = \sum_{j=0}^n x_{\sigma j} \beta_j + \epsilon_\sigma . \quad (54)$$

If y and ϵ are $nsam \times 1$ column vectors and β is an $(n+1) \times 1$ column vector, then can write previous equation in matrix form as:

$$y = X\beta + \epsilon . \quad (55)$$

Define the **projection matrices**

$$\wedge = X(X^T X)^{-1} X^T , \quad \vee = 1 - \wedge \quad (56)$$

A square matrix M is symmetric if $M^T = M$ and is idempotent if $M^2 = M$. \wedge is symmetric and idempotent and so is \vee . Note that \wedge and \vee also satisfy:

$$\vee \wedge = \wedge \vee = 0 \quad (57)$$

and

$$\wedge X = X , \quad \vee X = 0 . \quad (58)$$

One has

$$\beta = (X^T X)^{-1} X^T (y - \epsilon) . \quad (59)$$

Define

$$\boxed{\hat{\beta} = (X^T X)^{-1} X^T y = B y ,} \quad (60a)$$

$$\hat{y} = X \hat{\beta} = \wedge y , \quad (60b)$$

and

$$\hat{\epsilon} = y - X \hat{\beta} = y - \hat{y} = (1 - \wedge)y = \vee y . \quad (60c)$$

\wedge is sometimes called the **hat matrix**, because it gives y a hat.

Given any function $f = f(y, X, \epsilon)$ and a scalar factor $\xi \in \mathbb{R}$, suppose $f(\xi y, \xi X, \xi \epsilon) = \xi^{\mathcal{O}} f(y, X, \epsilon)$. Then we will say that $f(\cdot)$ is of **order \mathcal{O} under scaling**. Note that $\{X, y, \hat{y}, \epsilon, \hat{\epsilon}\}$ are all of order 1 under scaling, $\{\beta, \hat{\beta}, \wedge, \vee\}$ are all of order 0 under scaling, and B is of order -1 under scaling. Thus, the estimator variables (i.e, those with a hat) scale the same way as the variables without a hat that they are estimating. Furthermore, β , its estimator $\hat{\beta}$, and the projection matrices \wedge, \vee are invariant ($\mathcal{O} = 0$) under scaling.

Fig.3 illustrates that y can be expressed as a sum of 2 estimators:

$$y = \underbrace{\hat{y}}_{\wedge y} + \underbrace{\hat{\epsilon}}_{\vee y} . \quad (61)$$

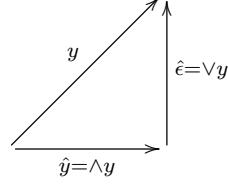


Figure 3: Decomposition of y into sum of two estimators, \hat{y} and $\hat{\epsilon}$.

model dependent results:

Assume the components of ϵ are random over σ and

$$E_\sigma[\epsilon] = \langle \epsilon \rangle = 0 \quad (62)$$

Assume X and β are not random. This makes $\underline{y} = X\beta + \underline{\epsilon}$ and $\hat{\beta} = (X^T X)^{-1} X^T \underline{y}$ random. One finds that

$$\langle \underline{y} \rangle = X\beta \quad (63)$$

$$\langle \hat{y} \rangle = \wedge \langle \underline{y} \rangle = \langle \underline{y} \rangle \quad (64a)$$

$$\langle \hat{\epsilon} \rangle = \vee \langle \underline{y} \rangle = 0 \quad (64b)$$

$$\langle \hat{\beta} \rangle = \beta \quad (64c)$$

So far, we have assumed a zero mean value for ϵ . Next, assume “**homoscedasticity**” (**HS**), which means that

$$\langle \underline{\epsilon}, \underline{\epsilon}^T \rangle = \xi^2 I_{nsam} \quad (64d)$$

where $\xi \geq 0$, $nsam = \sum_\sigma$ and I_{nsam} is the $nsam \times nsam$ identity matrix. It follows that

$$\langle \underline{y}, \underline{y}^T \rangle = \langle \underline{\epsilon}, \underline{\epsilon}^T \rangle = \xi^2 I_{nsam} , \quad (65)$$

$$\langle \hat{\epsilon}, \hat{\epsilon}^T \rangle = \vee \langle \underline{y}, \underline{y}^T \rangle \vee^T = \xi^2 \vee , \quad (66)$$

$$\langle \hat{y}, \hat{y}^T \rangle = \wedge \langle \underline{y}, \underline{y}^T \rangle \wedge^T = \xi^2 \wedge \quad (67)$$

and

$$\left\langle \hat{\beta}, \hat{\beta}^T \right\rangle = B \langle \underline{y}, \underline{y}^T \rangle B^T = \xi^2 (X^T X)^{-1}. \quad (68)$$

The goodness of fit for this model is often measured using the **coefficient of determination** R^2 . R^2 is defined by

$$R^2 = \frac{\| \hat{y} - \langle \hat{y} \rangle \|_2^2}{\| \underline{y} - \langle \underline{y} \rangle \|_2^2} = \frac{\text{tr} \langle \hat{y}, \hat{y}^T \rangle}{\text{tr} \langle \underline{y}, \underline{y}^T \rangle} \quad (69)$$

If HS holds, then R^2 reduces to

$$R^2 = \frac{\text{tr} \wedge}{nsam}. \quad (70)$$

0.21.2 LR, assuming x_σ are random and i.i.d.

Let

- $\underline{y} \in \mathbb{R}$: true value of dependent variable
- $\hat{y} \in \mathbb{R}$: estimator of dependent variable
- $\underline{\epsilon} \in \mathbb{R}$: residual
- $\underline{x}_j \in \mathbb{R}$: independent variables
- $\beta_0, \beta_j \in \mathbb{R}$: regression coefficients

$$\hat{y} = \beta_0 + \sum_{j=1}^n \beta_j \underline{x}_j \quad (71)$$

$$\underline{y} = \hat{y} + \underline{\epsilon} \quad (72)$$

Assume

$$\langle \underline{\epsilon} \rangle = 0 \quad (73)$$

and

$$\langle \underline{x}_j, \underline{\epsilon} \rangle = 0 \quad (74)$$

for all j .

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

$$\langle \underline{x}_k, \underline{y} \rangle = \sum_{j=1}^n \beta_j \langle \underline{x}_k, \underline{x}_j \rangle. \quad (75)$$

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

$$\langle \underline{x}^n, \underline{y} \rangle = \langle \underline{x}^n, (\underline{x}^n)^T \rangle \beta^n, \quad (76)$$

$$\boxed{\beta^n = \langle \underline{x}^n, (\underline{x}^n)^T \rangle^{-1} \langle \underline{x}^n, \underline{y} \rangle .} \quad (77)$$

$$\beta_0 = \langle \underline{y} \rangle - (\beta^n)^T \langle \underline{x}^n \rangle \quad (78)$$

Notice that the equations for the regression coefficients are very similar for the two cases of x_σ -nonrandom and x_σ -random. In fact, if we replace bilinears as follows

$$X^T y \longrightarrow \langle \underline{x}^n, \underline{y} \rangle \quad (79)$$

$$X^T X \longrightarrow \langle \underline{x}^n, (\underline{x}^n)^T \rangle \quad (80)$$

we go from the estimator of β for one case to the estimator of β for the other case:

$$\text{Eq.(60a)} \longrightarrow \text{Eq.(77)} \quad (81)$$

Next, we will write Eq.(77) for the special cases $n = 1$ and $n = 2$, where n is the number of independent variables \underline{x}_j

1. $n = 1$ (\underline{y} fitted by a line)

$$\underline{y} = \beta_0 + \beta \underline{x} + \epsilon \quad (82)$$

Eq.77 becomes

$$\beta_{\underline{y}, \underline{x}} = \beta = \frac{\langle \underline{y}, \underline{x} \rangle}{\langle \underline{x}, \underline{x} \rangle} \quad (83)$$

2. $n = 2$ (\underline{y} fitted by a plane)

$$\underline{y} = \beta_0 + \beta_1 \underline{x}_1 + \beta_2 \underline{x}_2 + \epsilon \quad (84)$$

Define

$$C_{i,j} = \langle \underline{x}_i, \underline{x}_j \rangle \quad (85)$$

for all i, j . Then Eq.77 becomes¹

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = C^{-1} \begin{bmatrix} \langle \underline{y}, \underline{x}_1 \rangle \\ \langle \underline{y}, \underline{x}_2 \rangle \end{bmatrix} \quad (86)$$

$$= \frac{1}{\det C} \begin{bmatrix} C_{22} & -C_{12} \\ -C_{21} & C_{11} \end{bmatrix} \begin{bmatrix} \langle \underline{y}, \underline{x}_1 \rangle \\ \langle \underline{y}, \underline{x}_2 \rangle \end{bmatrix} \quad (87)$$

¹ Recall that if $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ then $M^{-1} = \frac{1}{\det M} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

Hence,

$$\beta_{\underline{y}, \underline{x}_1 | \underline{x}_2} = \beta_1 = \frac{C_{22} \langle \underline{y}, \underline{x}_1 \rangle - C_{12} \langle \underline{y}, \underline{x}_2 \rangle}{C_{11} C_{12} - C_{12}^2} \quad (88)$$

Eq.(88) agrees with the value of $\beta_{YX|Z}$ in Ref.[22] by Pearl, if we replace in Pearl's formulae $X \rightarrow \underline{x}_1$, $Y \rightarrow \underline{y}$, $Z \rightarrow \underline{x}_2$.

0.22 Entropy, Kullback-Liebler divergence

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

- Entropy:

$$H(p) = - \sum_x p(x) \ln p(x) \geq 0 \quad (89)$$

- Kullback-Liebler divergence:

$$D_{KL}(p \parallel q) = \sum_x p(x) \ln \frac{p(x)}{q(x)} \geq 0 \quad (90)$$

- Cross entropy:

$$CE(p \rightarrow q) = - \sum_x p(x) \ln q(x) \quad (91)$$

$$= H(p) + D_{KL}(p \parallel q) \quad (92)$$

0.23 Definition of various entropies used in Shannon Information Theory

- (plain) Entropy of \underline{x}

$$H(\underline{x}) = - \sum_x P(x) \ln P(x) \quad (93)$$

This quantity measures the spread of $P_{\underline{x}}$. $H(\underline{x}) \geq 0$ and it vanishes iff $P(x) = \delta(x, x_0)$ (deterministic case)

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

$$H(\underline{y}|\underline{x}) = - \sum_{x,y} P(x,y) \ln P(y|x) \quad (94)$$

$$= H(\underline{y}, \underline{x}) - H(\underline{x}) \quad (95)$$

This quantity measures the conditional spread of \underline{y} given \underline{x} . $H(\underline{y}|\underline{x}) \geq 0$.

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

$$H(\underline{y} : \underline{x}) = \sum_{x,y} P(x,y) \ln \frac{P(x,y)}{P(x)P(y)} \quad (96)$$

$$= H(\underline{x}) + H(\underline{y}) - H(\underline{y}, \underline{x}) \quad (97)$$

This quantity measures the correlation between \underline{x} and \underline{y} . $H(\underline{y} : \underline{x}) \geq 0$ and it vanishes iff $P(x,y) = P(x)P(y)$.

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

$$H(\underline{y} : \underline{x} | \underline{\lambda}) = \sum_{x,y,\lambda} P(x,y,\lambda) \ln \frac{P(x,y|\lambda)}{P(x|\lambda)P(y|\lambda)} \quad (98)$$

$$= H(\underline{x} | \underline{\lambda}) + H(\underline{y} | \underline{\lambda}) - H(\underline{y}, \underline{x} | \underline{\lambda}) \quad (99)$$

This quantity measures the conditional correlation of \underline{x} and \underline{y} given $\underline{\lambda}$. $H(\underline{y} : \underline{x} | \underline{\lambda}) \geq 0$ and it vanishes iff $P(x,y|\lambda) = P(x|\lambda)P(y|\lambda)$.

An interesting special case occurs when $P(\lambda) = \delta(\lambda, \lambda_0)$ (the frequentist case of no λ prior.) In that case CMI reduces to

$$H(\underline{y} : \underline{x} | \lambda_0) = \sum_{x,y} P(x,y|\lambda_0) \ln \frac{P(x,y|\lambda_0)}{P(x|\lambda_0)P(y|\lambda_0)} \geq 0 \quad (100)$$

- 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

$$D_{KL}(P_{\underline{x}} \| P_{\underline{y}}) = \sum_x P_{\underline{x}}(x) \ln \frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} \quad (101)$$

This quantity measures a non-symmetric distance between the probability distributions $P_{\underline{x}}$ and $P_{\underline{y}}$. $D_{KL}(P_{\underline{x}} \| P_{\underline{y}}) \geq 0$ and it equals zero iff $P_{\underline{x}} = P_{\underline{y}}$.

0.24 Pearson Chi-Squared Test

The **Pearson divergence** (aka **Pearson Chi-squared test statistic**) for two probability distributions $P(x)$ and $Q(x)$, where $x \in S_{\underline{x}}$, is defined as follows:

$$D_{\chi^2} = \sum_x \frac{[P(x) - Q(x)]^2}{Q(x)} = \sum_x \frac{P^2(x)}{Q(x)} - 1. \quad (102)$$

²CMI can be read as “see me”.

As the following claim shows, the Pearson divergence is closely related to the Kullback-Liebler divergence.

Claim 2 If $\left| \frac{P(x)}{Q(x)} - 1 \right| << 1$ for all $x \in S_{\underline{x}}$, then

$$D_{KL}(P \parallel Q) \approx D_{\chi^2} . \quad (103)$$

proof:

$$D_{KL}(P \parallel Q) = \sum_x P(x) \ln \frac{P(x)}{Q(x)} \quad (104)$$

$$= \sum_x P(x) \ln \left(1 + \frac{P(x)}{Q(x)} - 1 \right) \quad (105)$$

$$\approx \sum_x P(x) \left(\frac{P(x)}{Q(x)} - 1 \right) \quad (106)$$

$$= \sum_x \frac{P^2(x)}{Q(x)} - 1 \quad (107)$$

$$= D_{\chi^2} \quad (108)$$

QED

Let $nx = |S_{\underline{x}}|$. Let $P_{\chi^2}(y)$ be the χ^2 (with $nx - 1$ degrees of freedom) probability distribution, and let $F_{\chi^2}(\alpha)$ be its cumulative distribution. Find α such that

$$95\% = \int_0^\alpha dy P_{\chi^2}(y) = F_{\chi^2}(\alpha) \quad (109)$$

If $D_{\chi^2} < \alpha$, then we say that $P = Q$ to 95% significance level (SL), whereas if $D_{\chi^2} > \alpha$, we say that $P \neq Q$ to 95% SL (i.e., SL=95%). The higher SL becomes, the higher α becomes, and the bigger the divergence D_{χ^2} has to be, before we are willing to declare that $P \neq Q$.

0.25 Demystifying Population and Sample Variances

Let $x[\sigma] = x^\sigma$. Given i.i.d.real variables $(x^\sigma)_{\sigma=0,1,\dots,n_{sam}-1}$, let

$$\hat{\mu} = \frac{1}{n_{sam}} \sum_\sigma x^\sigma \quad (110)$$

$$V_{partial}(\mu) = \frac{1}{n_{sam}} \sum_\sigma (x^\sigma - \mu)^2 \quad (111)$$

$$V_{full} = \frac{1}{nsam - 1} \sum_{\sigma} (\underline{x}^{\sigma} - \hat{\mu})^2 \quad (112)$$

Statisticians call $V_{partial}(\mu)$ the “population variance”. I will call it the **population variance for fixed μ** . Note that it depends on prior knowledge of the fixed parameter μ . Statisticians³ call V_{full} the “sample variance”. Instead, I will call V_{full} the **population variance for random μ** .

If one treats \underline{x}^{σ} as a random variable, then one must treat $\hat{\mu}$ as a random variable too. Let

$$E[\underline{x}^{\sigma}] = \mu \quad (113)$$

and

$$\left\langle \underline{x}^{\sigma}, \underline{x}^{\sigma'} \right\rangle = \delta(\sigma, \sigma') V_1 . \quad (114)$$

Then one can show that

$$E[V_{partial}(\mu)] = \frac{1}{nsam} E \left[\sum_{\sigma} (\underline{x}^{\sigma} - \mu)^2 \right] \quad (115)$$

$$= V_1 \quad (116)$$

and

$$E[V_{full}] = \frac{1}{nsam - 1} E \left[\sum_{\sigma} (\underline{x}^{\sigma} - \hat{\mu})^2 \right] \quad (117)$$

$$= V_1 \quad (118)$$

This is the reason why we use an $nsam - 1$ instead of an $nsam$ in V_{full} . Because it makes $E[V_{full}] = V_1$ so V_{full} is an unbiased estimator of the single individual variance V_1 . We say $\hat{\theta}$ is an **unbiased estimator** of a parameter θ if $E[\hat{\theta}] = \theta$.

The intuitive reason for why V_{full} is divided by $nsam - 1$ instead of $nsam$ is that whereas μ in $V_{partial}(\mu)$ is kept fixed and is “quiet”, the $\hat{\mu}$ in V_{full} is a random variable, noisy instead of quiet. The fluctuations in $\hat{\mu}$ are strongly correlated with the fluctuations of the \underline{x}^{σ} , so they decrease the fluctuations in V_{full} compared to those in $V_{partial}(\mu)$. By dividing by $nsam - 1$ instead of $nsam$, we compensate for this decrease in fluctuations so that the ratio of the numerator and denominator of V_{full} equals V_1 , instead of something *smaller* than V_1 , as would happen if were to divide by $nsam$

³ In their language, a “population” is supposed to be so large that its μ does not fluctuate, and a “sample” is supposed to be a small subset of that population for which the μ is assumed to fluctuate. In this book, I use the word “population” to mean a set of any size containing individuals, I use the word “sub-population” to refer to a subset of the population, and I use the word “sample” (aka individual, observation, unit, record) to mean a single individual of the population.

instead of $nsam - 1$. In terms of “degrees of freedom”(DOFs), $V_{partial}(\mu)$ has $nsam$ DOFs (namely one for each \underline{x}^σ), whereas V_{full} has $nsam - 1$ DOFs. (the presence of $\hat{\mu}$ subtracts one DOF). In both $V_{partial}(\mu)$ and V_{full} , one divides by the number of DOFs.

0.26 Error Bars

Never report measurements without error bars!! Quick reminder of error bars:

Normal distribution with mean μ and standard deviation σ :

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \quad (119)$$

Standard Normal distribution:

$$P(z) = \mathcal{N}(z; 0, 1) \quad (120)$$

Cumulative distribution for $P(z)$:

$$\Phi(z) = \int_{-\infty}^z dz' P(z') . \quad (121)$$

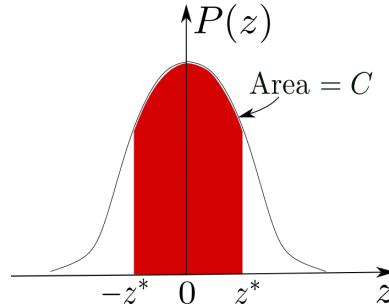


Figure 4: Interpretation of confidence level C .

Confidence Level C and corresponding z^* value (see Fig.4):

$$C = \int_{-z^*}^{z^*} dz P(z) = \Phi(z^*) - \Phi(-z^*) = 2 \left(\Phi(z^*) - \frac{1}{2} \right) \quad (122)$$

Equivalent definition:

$$C = P \left(\underbrace{\frac{|\underline{x} - \mu|}{\frac{\sigma}{\sqrt{n}}}}_{|z|} < z^* \right) \quad (123)$$

For $C = 95\%$, $z^* = 1.960 \approx 2$. For $C = 99\%$, $z^* = 2.576$.

Area of each tail in Fig.4 is usually called α :

$$C + 2\alpha = 1 \quad (124)$$

Estimators⁴ of mean μ and standard deviation σ from measurements x^σ of a sub-population Σ_1 of size $n = |\Sigma_1|$:

$$\hat{\mu} = \bar{x} = \frac{1}{n} \sum_{\sigma \in \Sigma_1} x^\sigma \quad (125)$$

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{\sigma \in \Sigma_1} (x^\sigma - \bar{x})^2 \quad (126)$$

We get from Eq.(123), the **Error bars (aka confidence interval)** and **Error E (aka margin of error)**:

$$\text{estimate of } x \text{ with error bars} = \bar{x} \pm z^* \underbrace{\frac{\hat{\sigma}}{\sqrt{n}}}_{E} \quad (127)$$

$$n = \left(\frac{z^* \hat{\sigma}}{E} \right)^2 \quad (128)$$

So far, we have assumed that the sub-population (aka sample population) is normally distributed. This might be false for several reasons. Some red flags: (1) n is too small (according to a rule of thumb derived from Central Limit Theorem, n should be bigger than 30 to insure normality). (2) Sub-population not truly random (i.i.d.) because was taken without replacement. In many cases, especially when $n < 30$, the Student's t-distribution models the sub-population statistics much better than the Normal distribution.

The **Student's t-distribution (SD)** $P_S(t; \nu = n-1)$, depends on a parameter ν called the number of degrees of freedom. In the case being considered here, ν equals the sub-population size n minus one. When fitting the data with SD, variable t replaces variable z , and $P_S(t; \nu = n-1)$ replaces the Standard Normal distribution (SND) $\mathcal{N}(z; \mu = 0, \sigma = 1)$. SD is symmetric about the origin like SND, but its tails are fatter. When fitting the data with SD, the z^* value is replaced by a t^* value. Eq.(122) is replaced by

$$C = \int_{-t^*}^{t^*} dt P_S(t) = \Phi_S(t^*) - \Phi_S(-t^*) = 2 \left(\Phi_S(t^*) - \frac{1}{2} \right) , \quad (129)$$

⁴Don't confuse the sample index σ with the standard deviation σ .

where quantities subscripted by S are for the SD, Also, Eq.(127) is replaced by

$$\text{estimate of } x \text{ with error bars} = \bar{x} \pm \underbrace{t^* \frac{\hat{\sigma}}{\sqrt{n}}}_{E} . \quad (130)$$

Tables of $t^*(C, \nu = n - 1)$ are available. Note that t^* depends on both C and ν , whereas $z^*(C)$ depends only on C .

0.27 Short Summary of Boolean Algebra

See Ref.[49] for more info about this topic.

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

$$x \text{ or } y = x \vee y = x + y - xy , \quad (131)$$

$$x \text{ and } y = x \wedge y = xy , \quad (132)$$

and

$$\text{not } x = \bar{x} = 1 - x , \quad (133)$$

where we are using normal addition and multiplication on the right hand sides.⁵

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

$$P(\bar{x}) = 1 - P(x) . \quad (134)$$

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

$$P(x \vee y) = P(x) + P(y) - P(x \wedge y) . \quad (135)$$

⁵Note 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	$x \wedge \bar{x} = 0$ $x \vee \bar{x} = 1$
Double negation	$\overline{(\bar{x})} = x$
De Morgan Laws	$\bar{x} \wedge \bar{y} = \overline{(x \vee y)}$ $\bar{x} \vee \bar{y} = \overline{(x \wedge y)}$

Table 1: Boolean Algebra Identities

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. 5 and 6 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 $j = 0, 1, 2, \dots, N - 1$ and $k = j - 1, j - 2, \dots, 0$.



Figure 5: Fully connected directed graph with 4 nodes, drawn as a line.

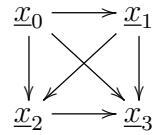


Figure 6: Fully connected directed graph with 4 nodes, drawn as a square.

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} .
- $de(\underline{a}) = \bigcup_{n=1}^{\infty} ch^n(\underline{a}) = ch(\underline{a}) \cup ch \circ ch(\underline{a}) \cup \dots$, descendants of \underline{a} .
- $an(\underline{a}) = \bigcup_{n=1}^{\infty} pa^n(\underline{a}) = pa(\underline{a}) \cup pa \circ pa(\underline{a}) \cup \dots$, ancestors of \underline{a} .

In this book, we will use \underline{a} . to indicate a **multi-node (node set, node array)** $\underline{a}. = (\underline{a}_j)_{j=0,1,\dots,na-1}$. 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 $\underline{a}.\cup\underline{b}.$, $\underline{a}.\cap\underline{b}.$, $\underline{a}.\setminus\underline{b}.$ 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, $\underline{a}.\setminus\underline{b} = \underline{a}.\setminus\{\underline{b}\}$.

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

$$P(x.) = \prod_j P(\underline{x}_j = x_j | (\underline{x}_k = x_k)_{k:\underline{x}_k \in pa(\underline{x}_j)}) . \quad (136)$$

Note that for a fully connected bnet with N nodes, Eq.(136) becomes

$$P(x.) = \prod_{j=0}^{N-1} P(x_j | (x_k)_{k=j-1,j-2,\dots,0}) . \quad (137)$$

For example, if $N = 4$, Eq.(137) becomes

$$P(x_0, x_1, x_2, x_3) = P(x_3 | x_2, x_1, x_0)P(x_2 | x_1, x_0)P(x_1 | x_0)P(x_0) . \quad (138)$$

We see that Eq.(137) is just the chain rule for conditional probabilities.

Given an arbitrarily large dataset of samples for the random variables $(\underline{x}_i)_{i=0,1,\dots,N-1}$, there may be several bnets (differing in the direction of some arrows) that fit the data well. However, according to Pearl’s causality theory, only one of these bnets is used

by Nature. I like to refer to that single one as the **causally correct (CC) Bayesian network**.⁶ In this book, whenever we speak of causal issues, we will assume, often without mentioning it, that the CC bnet is being used.⁷

⁶ The uniqueness of a CC bnet can be taken to be an implicit axiom of causality theory. Alternatively, instead of assuming uniqueness, one can assume that we are using a DAG which is, according to some measure of goodness of causal fit, the best one among the DAGs of a given set of DAGs. (see Chapter 22).

⁷We won't use the term "causal bnet" in this book. Pearl defines a causal bnet to be a CC bnet that is also a "SCM" (i.e., a bnet whose internal nodes are deterministic and external ones are probabilistic.)

Chapter 1

AdaBoost

This chapter is based on Ref.[44].

Adaptive Boosting (AdaBoost) is a method of constructing a strong classifier function as a linear combination of an ensemble of weak classifier functions.

Below, we will abbreviate “ensemble classifiers” by “e-classifiers” and “weak classifiers” by “w-classifiers”.

Chapter 12 defines decision trees (dtrees) and explains how to construct them. A **tree stump** is a dtree with only one parent and 2 children nodes. Usually the w-classifier functions for AdaBoost come from tree stumps (because tree stumps are w-classifiers and simple to compute), but the core AdaBoost algorithm is oblivious to where the w-classifier functions came from.

Chapter 45 is on Bagging (Random Forest), which is another method besides AdaBoosting of building a classifier function from an ensemble of classifier functions. These two methods are most commonly applied to dtrees: AdaBoosting for an ensemble of tree stumps, and Bagging for a random forest (which is an ensemble of dtrees that are usually much more complicated than tree stumps). Both methods are highly effective in mitigating overfitting, a common problem with simple dtrees.

1.1 AdaBoost for general ensemble of w-classifiers

In this section we discuss the core Adaboost algorithm, valid for a generic ensemble of w-classifiers.

Let $L = [0, 1, 2, \dots, nsam - 1]$ be a list of individuals (samples) in a population. In this chapter, we will use the notation $A^\sigma = A[\sigma]$ and $\vec{A} = [A^\sigma : \sigma \in L]$ for a list (vector, 1-D array) indexed by L . We will refer to $DS = (\vec{x}, \vec{y})$ where $x^\sigma \in S_{\underline{x}}$, $y^\sigma \in S_y$, as a dataset. Let $T = \{0, 1, \dots, nt - 1\}$. Let $x^\sigma = (x_0^\sigma, x_1^\sigma, \dots, x_{nt-1}^\sigma) \in S_{\underline{x}_0} \times S_{\underline{x}_1} \times \dots \times S_{\underline{x}_{nt-1}} = S_{\underline{x}}$.

AdaBoost assumes that the classifier class set S_y and all the feature sets $S_{\underline{x}_t}$ are binary: $S_y = S_{\underline{x}_t} = \{-1, 1\}$ for all $t \in T$.

Suppose that we are given an ensemble of nt **w-classifiers** $Y_t : S_{\underline{x}} \rightarrow \{-1, 1\}$, where $t \in T$. Suppose we want to find **intermediate e-classifiers** $\mathcal{Y}_t : S_{\underline{x}} \rightarrow \mathbb{R}$ given

by

$$\mathcal{Y}_t(x^\sigma) = \sum_{t'=0}^t \alpha_{t'} Y_{t'}(x^\sigma) \in \mathbb{R} \quad (1.1)$$

for $t \in T$ and a **final e-classifier** given by

$$\mathcal{Y}_{fin}(x^\sigma) = \text{sign}(\mathcal{Y}_{nt-1}(x^\sigma)) \in \{-1, 1\}. \quad (1.2)$$

The Adaboost algorithm yields a set of coefficients α_t for which the final e-classifier is much stronger (i.e., less error prone) than any of the w-classifiers of the ensemble.

Note that

$$\mathcal{Y}_t(x^\sigma) = \mathcal{Y}_{t-1}(x^\sigma) + \alpha_t Y_t(x^\sigma) \quad (1.3)$$

for $t \in T$ if we define $\mathcal{Y}_{-1} = 0$. Hence

$$\underbrace{e^{-y^\sigma \mathcal{Y}_t(x^\sigma)}}_{Z_t w_{t+1}^\sigma} = \underbrace{e^{-y^\sigma \mathcal{Y}_{t-1}(x^\sigma)}}_{Z_{t-1} w_t^\sigma} e^{-\alpha_t y^\sigma Y_t(x^\sigma)} \quad (1.4)$$

where the **weights** w_t^σ and the **partition function** Z_t are defined by

$$w_{t+1}^\sigma = \begin{cases} 1/nsam & \text{for } t = -1 \\ \frac{\exp(-y^\sigma \mathcal{Y}_t(x^\sigma))}{Z_t} & \text{for } t \geq 0 \end{cases} \quad (1.5)$$

and

$$Z_t = \sum_\sigma e^{-y^\sigma \mathcal{Y}_t(x^\sigma)}. \quad (1.6)$$

Note that the probability distribution $P(\sigma|t+1) = w_{t+1}^\sigma$ of weights at time $t+1$ emphasizes (i.e., gives higher probability to) the errors (i.e., occurrences of $y^\sigma \mathcal{Y}_t(x^\sigma) = -1$ for some population individual σ) of the previous (i.e., at time t) intermediate e-classifier \mathcal{Y}_t . In other words, every new intermediate e-classifier \mathcal{Y}_{t+1} concentrates on those individuals σ on which the previous e-classifier \mathcal{Y}_t performed poorly.

Note also that the partition function Z_t is a good measure of the **classification error** (i.e., occurrences of $y^\sigma \mathcal{Y}_t(x^\sigma) = -1$) of \mathcal{Y}_t . We will therefore use Z_t for that purpose, as an error measure.

For $t > 1$, we have

$$Z_t = \sum_\sigma e^{-y^\sigma \mathcal{Y}_t(x^\sigma)} \quad (1.7)$$

$$= \sum_\sigma \underbrace{e^{-y^\sigma \mathcal{Y}_{t-1}(x^\sigma)}}_{Z_{t-1} w_t^\sigma} e^{-\alpha_t y^\sigma Y_t(x^\sigma)} \quad (1.8)$$

$$= Z_{t-1} E_\sigma [e^{-\alpha_t y^\sigma Y_t(x^\sigma)}] \quad (1.9)$$

Define the **success rate** by

$$S_t = \sum_{\sigma} w_t^{\sigma} \mathbb{1}(y^{\sigma} Y_t(x^{\sigma}) = 1) \quad (1.10)$$

$$= E_{\sigma}[\underbrace{\mathbb{1}(y^{\sigma} Y_t(x^{\sigma}) = 1)}_{\text{iff } y^{\sigma} = Y_t(x^{\sigma})}] \quad (1.11)$$

and the **failure rate** by

$$F_t = \sum_{\sigma} w_t^{\sigma} \mathbb{1}(y^{\sigma} Y_t(x^{\sigma}) = -1) \quad (1.12)$$

$$= E_{\sigma}[\underbrace{\mathbb{1}(y^{\sigma} Y_t(x^{\sigma}) = -1)}_{\text{iff } y^{\sigma} \neq Y_t(x^{\sigma})}] . \quad (1.13)$$

Note that

$$S_t + F_t = 1 , \quad (1.14)$$

and

$$Z_t = Z_{t-1}(e^{-\alpha_t} S_t + e^{+\alpha_t} F_t) . \quad (1.15)$$

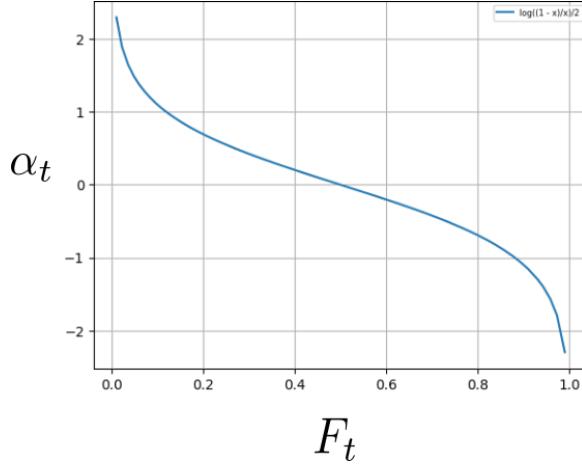


Figure 1.1: Plot of function $\alpha_t = \frac{1}{2} \ln \frac{1-F_t}{F_t}$.

We can find the α_t values that minimize the classification error Z_t , and then evaluate Z_t at those optimum α_t values:

$$\frac{dZ_t}{d\alpha_t} = Z_{t-1}(-e^{-\alpha_t} S_t + e^{\alpha_t} F_t) = 0 \quad (1.16)$$

$$e^{2\alpha_t} = \frac{S_t}{F_t} \quad (1.17)$$

$$\alpha_t = \frac{1}{2} \ln \frac{S_t}{F_t} = \frac{1}{2} \ln \frac{1 - F_t}{F_t} \quad (1.18)$$

$$\frac{Z_t}{Z_{t-1}} = 2\sqrt{S_t F_t} = 2\sqrt{(1 - F_t) F_t} \leq 1 \quad (1.19)$$

$f(x) = 2\sqrt{(1 - x)x}$ for $x \in [0, 1]$ is dome shaped and its maximum is 1, which is achieved iff $x = \frac{1}{2}$

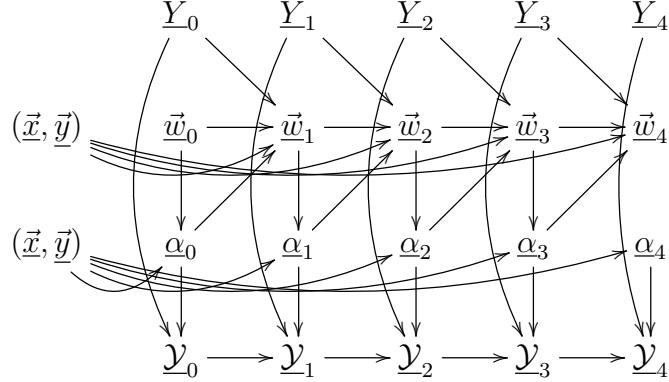


Figure 1.2: Bnet for AdaBoost with 5 w-classifiers, $nt = 5$. All nodes labelled (\vec{x}, \vec{y}) are the same node.

The AdaBoost algo described in this chapter is summarized by the bnet of Fig.1.2. The TPMs, printed in blue, for the nodes of that bnet, are as follows:

$$P(w_0^\sigma) = \frac{1}{nsam} \quad (1.20)$$

for all σ .

$$P(\vec{w}_{t+1} | \vec{w}_t, \alpha_t, Y_t, \vec{x}, \vec{y}) = \prod_\sigma \mathbb{1}(\vec{w}_{t+1}^\sigma = \frac{w_t^\sigma e^{-\alpha_t y^\sigma Y_t(x^\sigma)}}{\sum_\sigma \text{numerator}}) \quad (1.21)$$

for $t \geq 0$.

$$P(\alpha_t | \vec{w}_t, \vec{x}, \vec{y}) = \mathbb{1}(\alpha_t = \frac{1}{2} \ln \frac{1 - F_t}{F_t} \text{ where } F_t = F_t(\vec{w}_t, \vec{x}, \vec{y})) \quad (1.22)$$

for $t \in T$.

$$P(\mathcal{Y}_t | \mathcal{Y}_{t-1}, \alpha_t, Y_t) = \mathbb{1}(\mathcal{Y}_t = \mathcal{Y}_{t-1} + \alpha_t Y_t) \quad (1.23)$$

for $t \in T$, where $\mathcal{Y}_{-1} = 0$.

1.2 AdaBoost for ensemble of tree stumps

Keep in mind that AdaBoost assumes $S_y = S_{x_t} = \{-1, 1\}$ for all $t \in T = \{0, 1, \dots, nt - 1\}$. In order to implement AdaBoost, we need to specify nt w-classifiers $Y_t : \{-1, 1\}^{nt} \rightarrow \{-1, 1\}$ for $t \in T$. One can either specify the nt w-classifiers a priori or build them on-the-fly.

- **w-classifiers specified a priori**

Define a classifier for each feature x_t where $t \in T$ by:

$$Y_t(x^\sigma) = x_t^\sigma \in \{-1, 1\} \quad (1.24)$$

Hence, for this classifier, $y^\sigma Y_t(x^\sigma) = y^\sigma x_t^\sigma$.

- **w-classifiers built on-the-fly**

Recall dataset $(\vec{x}, \vec{y}) = [(x^\sigma, y^\sigma) : \sigma \in L]$ is indexed by the list $L = [0, 1, \dots, nsam - 1]$. If L_j is a list (possibly with duplicate items) such that $set(L_j) \subset set(L)$, then define $DS_j = (\vec{x}, \vec{y})_{L_j} = ((x^\sigma)_{\sigma \in L_j}, (y^\sigma)_{\sigma \in L_j})$. We will refer to DS_j as the **restriction of (\vec{x}, \vec{y}) to L_j** .

The idea behind on-the-fly w-classifiers is to choose $Y_t(x^\sigma) = x_t^\sigma$, where x_t is the feature with the lowest Gini in the current dataset $(\vec{x}, \vec{y})_{L_t}$. To build $(\vec{x}, \vec{y})_{L_t}$, we select at random from $L = [0, 1, \dots, nsam - 1]$, a list L_t of the same length as L , using the probability distribution \vec{w}_{t-1} . By choosing L_t with probabilities \vec{w}_{t-1} , we emphasize individuals σ that are failing. Then we define $(\vec{x}, \vec{y})_{L_t}$ as the restriction of (\vec{x}, \vec{y}) to L_t .

Perhaps a causal diagram will make all these new steps clearer to the reader. The bnet Fig.1.3 is a modification of the bnet Fig.1.2 to include these new steps. The TPMs, printed in blue, for new or changed nodes, are as follows:

$$P(Y_t | (\vec{x}, \vec{y})_{L_t}) = \mathbb{1}(\text{ } Y_t(x^\sigma) = x_t^\sigma \text{ where } x_t \text{ is feature in } (\vec{x}, \vec{y})_{L_t} \text{ the with lowest Gini. }) \quad (1.25)$$

$$P(L_{t+1}^\sigma = \sigma' | \vec{w}_t) = w_t^{\sigma'} \quad (1.26)$$

$$P((\vec{x}, \vec{y})_{L_t} | L_t, (\vec{x}, \vec{y})) = \mathbb{1}(\text{ } (\vec{x}, \vec{y})_{L_t} = \text{ restriction of } (\vec{x}, \vec{y}) \text{ to } L_t \text{ }) \quad (1.27)$$

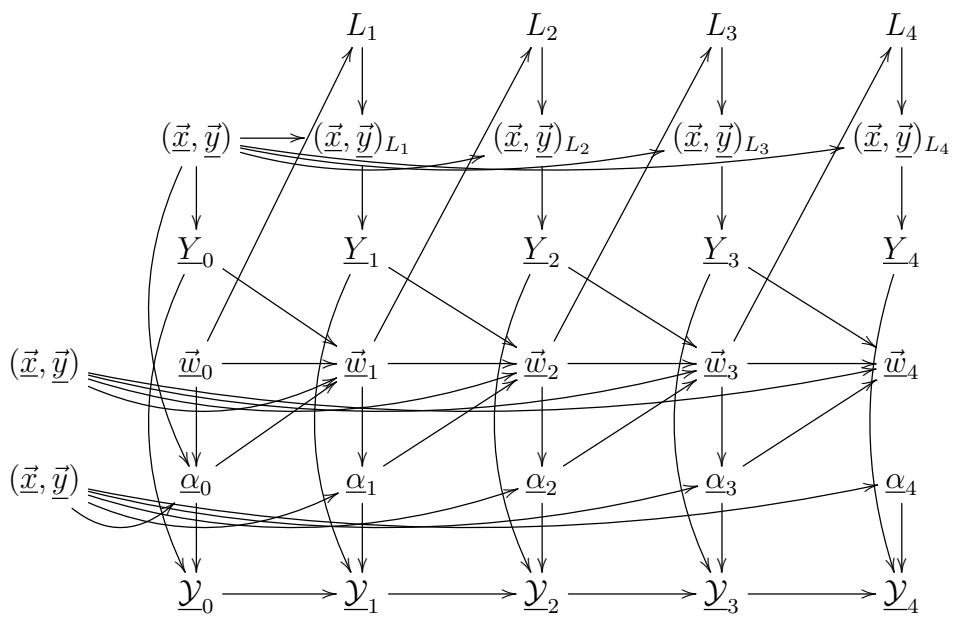


Figure 1.3: Modification of the bnet of Fig.1.2 to include on-the-fly generation of the w-classifiers.

Chapter 2

ARACNE structure learning

This chapter is based on Ref.[17].

The ARACNE algo is an algo for learning the structure of a bnet from data. The algo considers data samples for n random variables $(x_i)_{i=0,1,\dots,n-1}$, 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 $0 < \epsilon \ll 1$. 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 54).

Ref.[17] 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 34 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.

Fig.2.1 gives an example of the application of the ARACNE algo.

See Chapter 9 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.2.1 (a) was considered in Chapter 9 on CLTs, and the CLT algo also gave Fig.2.1 (c) as the final structure.

According to Ref.[17], 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

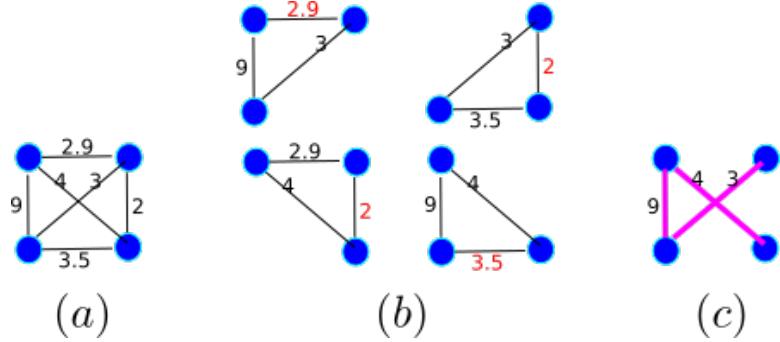


Figure 2.1: An example where the ARACNE algo gives a Chow-Liu tree. (a) Fully connected undirected graph with weights $MI_{i,j}$ along the edges. (b) All 4 possible triplets of edges with nonzero weights. Edges marked for removal have their weights printed in red.(c) Final structure.

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 3

Backdoor Adjustment

The backdoor (BD) adjustment theorem is proven in Chapter 15 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 15 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 16.

Suppose that 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 the 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 backdoor paths from $\underline{x}.$ to $\underline{y}.$ are blocked by $\underline{z}..$
2. $\underline{z}.. \cap de(\underline{x}.) = \emptyset.$

Claim 3 Backdoor Adjustment Theorem

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

$$P(y.| \rho \underline{x}. = x.) = \sum_{z.} P(y.| x., z.) P(z.) \quad (3.1)$$

$$= \sum_{z.} \begin{array}{c} \underline{z}.. = z. \\ \searrow \\ \underline{x}.. = x. \longrightarrow \underline{y}.. \end{array} \quad (3.2)$$

proof: See Chapter 15

QED

3.1 Examples

1.

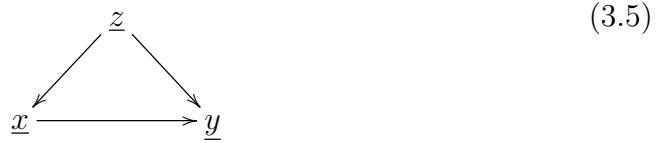


(3.3)

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

$$P(y|\rho \underline{x} = x) = P(y|x) \quad (3.4)$$

2.



(3.5)

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.



(3.6)

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

4.



(3.7)

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 19.

5.



(3.8)

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 w of \underline{x} because it is hidden, but we are able to block the backdoor path by conditioning on \underline{z} instead.

6.



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.



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

$$P(y|\rho\underline{x} = x) = P(y|x) . \quad (3.11)$$

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,

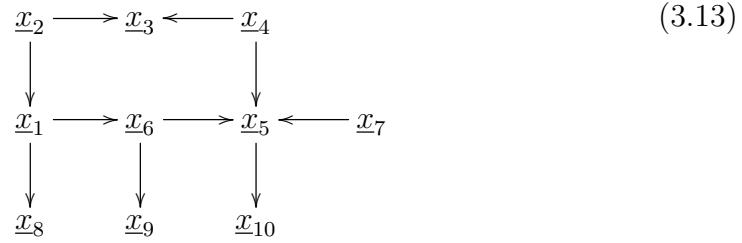
$$P(y|\rho\underline{x} = x) = \sum_t P(y|x, t)P(t) . \quad (3.12)$$

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.[11])

Consider the bnet



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

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

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

Chapter 4

Back Propagation (Automatic Differentiation)

4.1 General Theory

4.1.1 Jacobians

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

$$y = f(x) . \quad (4.1)$$

Then the Jacobian $\frac{\partial y}{\partial x}$ is defined as the matrix with entries¹

$$\left[\frac{\partial y}{\partial x} \right]_{i,j} = \frac{\partial y_i}{\partial x_j} . \quad (4.2)$$

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

$$y = g \circ f(x) , \quad (4.3)$$

then

$$\frac{\partial y}{\partial x} = \frac{\partial g}{\partial f} \frac{\partial f}{\partial x} . \quad (4.4)$$

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

$$\underline{f}^4 \longleftarrow \underline{f}^3 \longleftarrow \underline{f}^2 \longleftarrow \underline{f}^1 \longleftarrow \underline{f}^0$$

(a) Composition

$$\frac{\partial f^4}{\partial x} \longleftarrow \frac{\partial f^3}{\partial x} \longleftarrow \frac{\partial f^2}{\partial x} \longleftarrow \frac{\partial f^1}{\partial x} \longleftarrow 1$$

(b) Forward-p

$$1 \longrightarrow \frac{\partial y}{\partial f^3} \longrightarrow \frac{\partial y}{\partial f^2} \longrightarrow \frac{\partial y}{\partial f^1} \longrightarrow \frac{\partial y}{\partial f^0}$$

(c) Back-p

Figure 4.1: bnets for function composition, forward propagation and back propagation for $nf = 5$ nodes.

4.1.2 bnets for function composition, forward propagation and back propagation

Let

$$y = f^4 \circ f^3 \circ f^2 \circ f^1(x) . \quad (4.5)$$

This function composition chain can be represented by the bnet Fig.4.1(a) with TPMs

$$P(f^\mu | f^{\mu-1}) = \mathbb{1}(f^\mu = f^\mu(f^{\mu-1})) \quad (4.6)$$

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.

Note that

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial f^3} \frac{\partial f^3}{\partial f^2} \left[\frac{\partial f^2}{\partial f^1} \frac{\partial f^1}{\partial x} \right] \quad (4.7)$$

$$= \frac{\partial y}{\partial f^3} \left[\frac{\partial f^3}{\partial f^2} \frac{\partial f^2}{\partial x} \right] \quad (4.8)$$

$$= \left[\frac{\partial y}{\partial f^3} \frac{\partial f^3}{\partial x} \right] \quad (4.9)$$

$$= \frac{\partial y}{\partial x}. \quad (4.10)$$

This forward propagation can be represented by the bnet Fig.4.1(b) with node TPMs

$$P\left(\frac{\partial f^{\mu+1}}{\partial x} \mid \frac{\partial f^\mu}{\partial x}\right) = \mathbb{1}\left(\frac{\partial f^{\mu+1}}{\partial x} = \frac{\partial f^{\mu+1}}{\partial f^\mu} \frac{\partial f^\mu}{\partial x}\right) \quad (4.11)$$

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

Note that

$$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial f^3} \frac{\partial f^3}{\partial f^2} \right] \frac{\partial f^2}{\partial f^1} \frac{\partial f^1}{\partial x} \quad (4.12)$$

$$= \left[\frac{\partial y}{\partial f^2} \frac{\partial f^2}{\partial f^1} \right] \frac{\partial f^1}{\partial x} \quad (4.13)$$

$$= \left[\frac{\partial y}{\partial f^1} \frac{\partial f^1}{\partial x} \right] \quad (4.14)$$

$$= \frac{\partial y}{\partial x}. \quad (4.15)$$

This back propagation can be represented by the bnet Fig.4.1(c) with node TPMs

$$P\left(\frac{\partial y}{\partial f^\mu} \mid \frac{\partial y}{\partial f^{\mu+1}}\right) = \mathbb{1}\left(\frac{\partial y}{\partial f^\mu} = \frac{\partial y}{\partial f^{\mu+1}} \frac{\partial f^{\mu+1}}{\partial f^\mu}\right) \quad (4.16)$$

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.

4.2 Application to Neural Networks

4.2.1 Absorbing b_i^λ into $w_{i|j}$.

Below are, printed in blue, the TPMs for the nodes of a NN bnet, as given in Chapter 39.

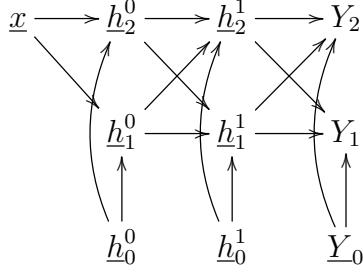


Figure 4.2: Nodes $\underline{h}_0^0, \underline{h}_0^1, \underline{Y}_0$ are all set to 1. They allow us to absorb b_i^λ into the first column of $w_{i|j}^\lambda$.

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

$$P(h_i^\lambda | h_{\cdot}^{\lambda-1}) = \delta \left(h_i^\lambda, \mathcal{A}_i^\lambda \left(\sum_j w_{i|j}^\lambda h_j^{\lambda-1} + b_i^\lambda \right) \right) \quad (4.17)$$

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

$$P(Y_i | h_{\cdot}^{\Lambda-2}) = \delta \left(Y_i, \mathcal{A}_i^{\Lambda-1} \left(\sum_j w_{i|j}^{\Lambda-1} h_j^{\Lambda-2} + b_i^{\Lambda-1} \right) \right) \quad (4.18)$$

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

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

$$w_{i|0}^\lambda = b_i^\lambda \quad (4.19)$$

Let the nodes \underline{h}_0^λ 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^λ 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 $\lambda = 0, 1, \dots, \Lambda - 2$,

$$P(h_0^\lambda) = \delta(h_0^\lambda, 1) , \quad (4.20)$$

and

$$P(h_i^\lambda | h_{\cdot}^{\lambda-1}, h_0^\lambda = 1) = \delta \left(h_i^\lambda, \mathcal{A}_i^\lambda \left(\sum_j w_{i|j}^\lambda h_j^{\lambda-1} \right) \right) \quad (4.21)$$

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

$$P(Y_0) = \delta(Y_0, 1) , \quad (4.22)$$

and

$$P(Y_i \mid h_{\cdot}^{\Lambda-2}, Y_0 = 1) = \delta \left(Y_i, \mathcal{A}_i^{\Lambda-1} \left(\sum_j w_{i|j}^{\Lambda-1} h_j^{\Lambda-2} \right) \right) \quad (4.23)$$

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

4.2.2 bnets for function composition, forward propagation and back propagation for NN

$$\underline{\mathcal{A}}^3 \longleftarrow \underline{\mathcal{B}}^3 \longleftarrow \underline{\mathcal{A}}^2 \longleftarrow \underline{\mathcal{B}}^2 \longleftarrow \underline{\mathcal{A}}^1 \longleftarrow \underline{\mathcal{B}}^1 \longleftarrow \underline{\mathcal{A}}^0 \longleftarrow \underline{\mathcal{B}}^0 \longleftarrow \underline{x}$$

(a)

$$\underline{\frac{\partial \mathcal{A}^3}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{B}^3}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{A}^2}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{B}^2}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{A}^1}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{B}^1}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{A}^0}{\partial x}} \longleftarrow \underline{\frac{\partial \mathcal{B}^0}{\partial x}} \longleftarrow \underline{1}$$

(b)

$$\underline{1} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{B}^3}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{A}^2}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{B}^2}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{A}^1}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{B}^1}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{A}^0}} \longrightarrow \underline{\frac{\partial Y}{\partial \mathcal{B}^0}} \longrightarrow \underline{\frac{\partial Y}{\partial x}}$$

(c)

Figure 4.3: bnets for (a) function composition, (b) forward propagation and (c) back propagation for a neural net with 4 layers (3 hidden and output visible).

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

$$\mathcal{E} = \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} |Y_i - y_i[\sigma]|^2 \quad (4.24)$$

To perform simple gradient descent, one uses:

$$(w_{i|j}^\lambda)' = w_{i|j}^\lambda - \eta \frac{\partial \mathcal{E}}{\partial w_{i|j}^\lambda} . \quad (4.25)$$

One has

$$\frac{\partial \mathcal{E}}{\partial w_{i|j}^\lambda} = \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} 2(Y_i - y_i[\sigma]) \frac{\partial Y}{\partial w_{i|j}^\lambda}. \quad (4.26)$$

Define \mathcal{B}_i^λ thus

$$\mathcal{B}_i^\lambda(h^{\lambda-1}) = \sum_j w_{i|j}^\lambda h_j^{\lambda-1}. \quad (4.27)$$

Then

$$\frac{\partial Y}{\partial w_{i|j}^\lambda} = \frac{\partial Y}{\partial \mathcal{B}_i^\lambda} \frac{\partial \mathcal{B}_i^\lambda}{\partial w_{i|j}^\lambda} \quad (4.28)$$

$$= \frac{\partial Y}{\partial \mathcal{B}_i^\lambda} h_j^{\lambda-1} \quad (4.29)$$

$$\frac{\partial \mathcal{E}}{\partial w_{i|j}^\lambda} = \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} \frac{\partial \mathcal{B}_j^\lambda}{\partial w_{i|j}^\lambda} \quad (4.30)$$

$$= \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} h_j^{\lambda-1}. \quad (4.31)$$

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

$$\begin{cases} \delta_j^\lambda = \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} \\ \frac{\partial \mathcal{E}}{\partial w_{i|j}^\lambda} = \delta_j^\lambda h_j^{\lambda-1} \end{cases} \quad (4.32)$$

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:

$$f_i^0 = x_i \quad (4.33)$$

$$\text{Layer 0: } f_i^1 = \mathcal{B}_i^0(x_i), \quad f_i^2 = \mathcal{A}_i^0(\mathcal{B}_i^0) \quad (4.34)$$

$$\text{Layer 1: } f_i^3 = \mathcal{B}_i^1(\mathcal{A}_i^0), \quad f_i^4 = \mathcal{A}_i^1(\mathcal{B}_i^1) \quad (4.35)$$

$$\text{Layer 2: } f_i^5 = \mathcal{B}_i^2(\mathcal{A}_i^1), \quad f_i^6 = \mathcal{A}_i^2(\mathcal{B}_i^2) \quad (4.36)$$

$$\text{Layer 3: } f_i^7 = \mathcal{B}_i^3(\mathcal{A}_i^2), \quad f_i^8 = \mathcal{A}_i^3(\mathcal{B}_i^3) \quad (4.37)$$

See Fig.4.3. The TPMs, printed in blue, for the nodes of the bnet (*c*) for back propagation, are:

$$P\left(\frac{\partial Y}{\partial \mathcal{B}^\lambda} \mid \frac{\partial Y}{\partial \mathcal{B}^{\lambda+1}}\right) = \mathbb{1}\left(\frac{\partial Y}{\partial \mathcal{B}^\lambda} = \frac{\partial Y}{\partial \mathcal{B}^{\lambda+1}} \frac{\partial \mathcal{B}^{\lambda+1}}{\partial \mathcal{A}^\lambda} \frac{\partial \mathcal{A}^\lambda}{\partial \mathcal{B}^\lambda}\right). \quad (4.38)$$

One has

$$\frac{\partial \mathcal{A}_i^\lambda}{\partial \mathcal{B}_j^\lambda} = D\mathcal{A}_i^\lambda(\mathcal{B}_i^\lambda)\delta(i, j) \quad (4.39)$$

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

From Eq.(4.27)

$$\mathcal{B}_i^{\lambda+1}(\mathcal{A}^\lambda) = \sum_j w_{i|j}^{\lambda+1} \mathcal{A}_j^\lambda \quad (4.40)$$

so

$$\frac{\partial \mathcal{B}_i^{\lambda+1}}{\partial \mathcal{A}_j^\lambda} = w_{i|j}^{\lambda+1}. \quad (4.41)$$

Therefore, Eq.(4.38) implies

$$P\left(\frac{\partial Y}{\partial \mathcal{B}_j^\lambda} \mid \frac{\partial Y}{\partial \mathcal{B}_j^{\lambda+1}}\right) = \mathbb{1}\left(\frac{\partial Y}{\partial \mathcal{B}_j^\lambda} = \sum_i \frac{\partial Y}{\partial \mathcal{B}_i^{\lambda+1}} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda) w_{i|j}^{\lambda+1}\right), \quad (4.42)$$

$$P\left(\frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} \mid \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^{\lambda+1}}\right) = \mathbb{1}\left(\frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^\lambda} = \sum_i \frac{\partial \mathcal{E}}{\partial \mathcal{B}_i^{\lambda+1}} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda) w_{i|j}^{\lambda+1}\right), \quad (4.43)$$

$$P(\delta_j^\lambda \mid \delta_j^{\lambda+1}) = \mathbb{1}(\delta_j^\lambda = \sum_i \delta_i^{\lambda+1} D\mathcal{A}_j^\lambda(\mathcal{B}_j^\lambda) w_{i|j}^{\lambda+1}). \quad (4.44)$$

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

$$\delta_j^{\Lambda-1} = \frac{\partial \mathcal{E}}{\partial \mathcal{B}_j^{\Lambda-1}} \quad (4.45)$$

$$= \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} \sum_{i=0}^{ny-1} 2(Y_i - y_i[\sigma]) D\mathcal{A}_i^{\Lambda-1}(\mathcal{B}_i^{\Lambda-1}) \delta(i, j) \quad (4.46)$$

$$= \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} 2(Y_j - y_j[\sigma]) D\mathcal{A}_j^{\Lambda-1}(\mathcal{B}_j^{\Lambda-1}) \quad (4.47)$$

Cute expression for derivative of sigmoid function:

$$Dsmoid(x) = smoid(x)(1 - smoid(x)) \quad (4.48)$$

4.3 General bnets instead of Markov chains induced by layered structure of NNs

$$P(\delta_{\underline{x}} \mid (\delta_{\underline{a}})_{\underline{a} \in ch(\underline{x})}) = \mathbb{1}(\delta_{\underline{x}} = \sum_{\underline{a} \in ch(\underline{x})} \delta_{\underline{a}} D \mathcal{A}_{\underline{x}}(\mathcal{B}_{\underline{x}})) w_{\underline{a}|\underline{x}}) \quad (4.49)$$

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

$$P(\delta_{\underline{x}} \mid (\delta_{\underline{a}})_{\underline{a} \in pa(\underline{x})}) = \mathbb{1}(\delta_{\underline{x}} = \sum_{\underline{a} \in pa(\underline{x})} \delta_{\underline{a}} D \mathcal{A}_{\underline{x}}(\mathcal{B}_{\underline{x}})) w_{\underline{x}|\underline{a}}^T) \quad (4.50)$$

Chapter 5

Basic Curve Fitting Using Gradient Descent

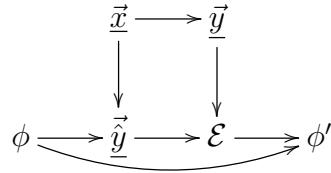


Figure 5.1: Basic curve fitting bnet.

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

$$P_{\underline{x}, \underline{y}}(x, y) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x = x[\sigma], y = y[\sigma]). \quad (5.1)$$

Let

$$\mathcal{E}(\vec{x}, \vec{y}, \phi) = \frac{1}{nsam(\vec{y})} \sum_{\sigma} |y[\sigma] - \hat{y}(x[\sigma]; \phi)|^2 \quad (5.2)$$

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

Best fit is parameters ϕ^* such that

$$\phi^* = \operatorname{argmin}_{\phi} \mathcal{E}(\vec{x}, \vec{y}, \phi). \quad (5.3)$$

The node TPMs for the basic curve fitting bnet Fig.5.1 are printed below in blue.

$$P(\phi) = \text{given}. \quad (5.4)$$

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

$$P(\vec{x}) = \prod_{\sigma} P_{\underline{x}}(x[\sigma]) \quad (5.5)$$

$$P(\vec{y}|\vec{x}) = \prod_{\sigma} P_{\underline{y}|\underline{x}}(y[\sigma] | x[\sigma]) \quad (5.6)$$

$$P(\hat{y}[\sigma]|\phi, \vec{x}) = \delta(\hat{y}[\sigma], \hat{y}(x[\sigma]; \phi)) \quad (5.7)$$

$$P(\mathcal{E}|\vec{y}, \vec{x}) = \delta(\mathcal{E}, \frac{1}{nsam(\vec{x})} \sum_{\sigma} |y[\sigma] - \hat{y}[\sigma]|^2) . \quad (5.8)$$

$$P(\phi'|\phi, \mathcal{E}) = \delta(\phi', \phi - \eta \partial_{\phi} \mathcal{E}) \quad (5.9)$$

$\eta > 0$ 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 6

Bell and Clauser-Horne Inequalities in Quantum Mechanics

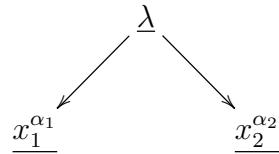


Figure 6.1: bnet used to discuss Bell and Clauser-Horne inequalities in Quantum Mechanics.

I wrote an article about this in 2008 for my blog “Quantum Bayesian Networks”. See Ref.[40].

Chapter 7

Berkson's Paradox

For more information about Berkson's Paradox (BP), see Ref.[46]

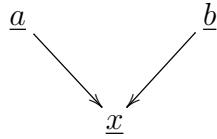


Figure 7.1: Bnet used to discuss Berkson's Paradox (BP). \underline{a} and \underline{b} are common causes of collider \underline{x} .

Consider the bnet of Fig.7.1. For that bnet, we have

$$P(a, b, x) = P(a)P(b)P(x|a, b) . \quad (7.1)$$

Summing Eq.(7.1) over x , we get

$$P(a, b) = P(a)P(b) \quad (7.2)$$

so \underline{a} and \underline{b} are independent. It follows that a can be ignored in calculating the probability of b ; i.e.,

$$\boxed{P(b|a) = P(b)} . \quad (7.3)$$

However, a cannot be ignored in calculating the probability of b , if x is being held fixed; i.e.,

$$\boxed{P(b|a, x) \neq P(b|x)} . \quad (7.4)$$

Indeed,

$$P(b|a, x) = \frac{P(b)P(x|a, b)}{\sum_b P(b)P(x|a, b)} \quad (7.5)$$

whereas

$$P(b|x) = \frac{\sum_a P(a)P(b)P(x|a,b)}{\sum_{a,b} P(a)P(b)P(x|a,b)} . \quad (7.6)$$

The two boxed equations are what is referred to as BP.

BP is also called **collider bias** because x is a collider.

BP is also called **explaining away** in the special case that $a, b, x \in \{true, false\} = \{0, 1\}$. In that case, if x is fixed to true, and the cause a is known to be true, then the cause b is less likely to be true. For example, suppose a car engine fails ($x = 1$) and the two most likely causes of the failure are alternator (a) and battery (b). Once we know that the alternator has failed ($a = 1$), it is less likely that the battery is failing ($b = 1$) than when the status of a was not known; i.e., $P(b = 1|x = 1, a = 1) < P(b = 1|x = 1)$.

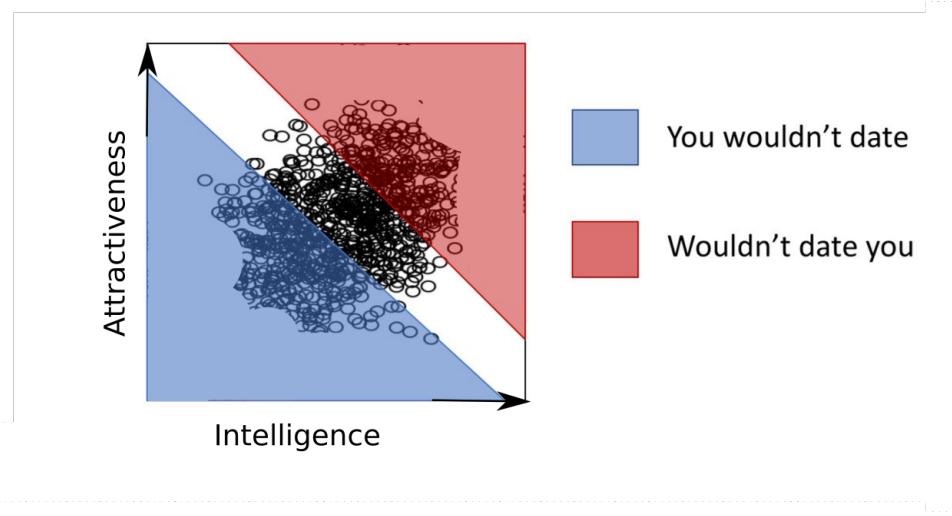


Figure 7.2: Example of Berkson's paradox (BP).

Fig.7.2 presents an example of BP. The figure consists of a scatter plot with axes $x=$ intelligence, $y=$ attractiveness, for a population of possible dates for a single person. For the full population,

$$(a, b) \sim P(a, b) = P(a)P(b) \quad (7.7)$$

whereas for the population in the white swath,

$$(a, b) \sim P(a, b|x) = P(b|a, x)P(a|x) \neq P(b|x)P(a|x) . \quad (7.8)$$

As shown by Fig.7.2, BP is an example of **selection bias**. Selection bias happens when a non-representative subset of the total population is considered (i.e., selected).

Chapter 8

Binary Decision Diagrams

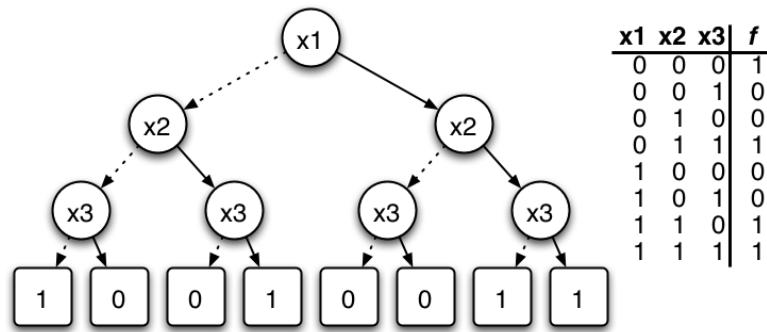


Figure 8.1: Binary decision tree and truth table for the function $f(x_1, x_2, x_3) = \bar{x}_1(x_2 + \bar{x}_3) + x_1x_2$

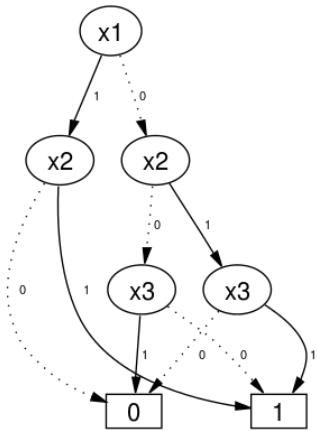


Figure 8.2: BDD for the function f of Fig.8.1.

This chapter is based on Wikipedia article Ref.[48].

Binary Decision Diagrams (BDDs) can be understood as a special case of Decision Trees (dtrees). We will assume that the reader has read Chapter 12 on dtrees before reading this chapter.

Both Figs.8.1 and 8.2 were taken from the aforementioned Wikipedia article. They give a simple example of a function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ represented in Fig.8.1 as a **binary decision tree** and in Fig.8.2 as a **binary decision diagram (BDD)**. It is possible to find, for each of those two figures, a bnet with the same graph structure. We show how to do this next.

We begin by noting that the function $f : \{0, 1\}^3 \rightarrow \{0, 1\}$ is a special case of a probability distribution $P : \{0, 1\}^3 \rightarrow [0, 1]$. In fact, if we restrict P to be deterministic, then $P_{det} : \{0, 1\}^3 \rightarrow \{0, 1\}$ 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

$$P(x_1, x_2, x_3) = P(x_3|x_1, x_2)P(x_2|x_1)P(x_1), \quad (8.1)$$

which can be represented by the bnet:

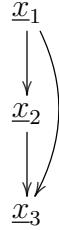


Figure 8.3: Most general 3 node bnet.

But in Chapter 12, we learned how to represent the dtree of Fig.8.1 as the image bnet Fig.8.4. The node TPMs, printed in blue, for the image bnet Fig.8.4 are as follows. Note that the TPMs for Fig.8.4 can be constructed from the TPMs for the bnet Fig.8.3. If $x_1, x_2, x_3, x_4 \in \{0, 1, null\}$ and $a, b, c \in \{0, 1\}$, then

$$P(\underline{x}_1 = x_1) = \begin{cases} P_{x_1}(x_1) & \text{if } x_1 \in \{0, 1\} \\ 0 & \text{if } x_1 = \text{null} \end{cases} \quad (8.2)$$

$$P(\underline{x}_2|a = x_2 | \underline{x}_1 = x_1) = \begin{cases} P_{x_2|x_1}(x_2|a) & \text{if } x_1 = a \\ \mathbb{1}(x_2 = \text{null}) & \text{otherwise} \end{cases} \quad (8.3)$$

$$P(\underline{x}_3|a, b = x_3 | \underline{x}_2|a = x_2) = \begin{cases} P_{x_3|x_1, x_2}(x_3|a, b) & \text{if } x_2 = b \\ \mathbb{1}(x_3 = \text{null}) & \text{otherwise} \end{cases} \quad (8.4)$$

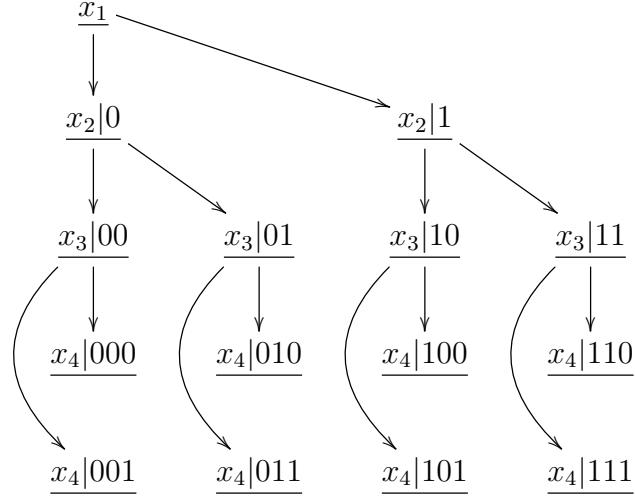


Figure 8.4: Image bnet for binary dtree of Fig.8.1.

$$P(\underline{x}_4|a, b, c = x_4 | \underline{x}_3|a, b = x_3) = \begin{cases} \delta(x_4, c) & \text{if } x_3 = c \\ \mathbb{1}(x_4 = \text{null}) & \text{otherwise} \end{cases} \quad (8.5)$$

Note that if $P_{\underline{x}_3|x_1, x_2} = P_{\underline{x}_3|x_2}$ in Eq.(8.4), then the bnet Fig.8.3 reduces to a Markov chain $\underline{x}_1 \rightarrow \underline{x}_2 \rightarrow \underline{x}_3$.

The BDD shown in Fig.8.2 emphasizes the fact that

$$P(x_1, x_2, x_3|x_1 = 1) = P(x_2|x_1 = 1) = x_2 . \quad (8.6)$$

The BDD of Fig.8.2 has as image bnet Fig.8.5. Define

$$pa(\underline{0}) = pa(\underline{1}) = (x_2|1, x_3|00, x_3|01) . \quad (8.7)$$

Let $pa(\underline{0}) = abc$ mean the same as $pa(\underline{0}) = (a, b, c)$. The TPMs of the image bnet Fig.8.5 are the same as those for the image bnet Fig.8.4 except for the TPMs of the nodes $\underline{0}$ and $\underline{1}$. For those two nodes, the TPMs, printed in blue, are as follows.

$$P(\underline{0} = x|pa(\underline{0})) = \begin{cases} \delta(x, 0) & \text{if } pa(\underline{0}) = 011 \\ \delta(x, \text{null}) & \text{otherwise} \end{cases} \quad (8.8)$$

$$P(\underline{1} = x|pa(\underline{1})) = \begin{cases} \delta(x, 1) & \text{if } pa(\underline{1}) = 101 \\ \delta(x, \text{null}) & \text{otherwise} \end{cases} \quad (8.9)$$

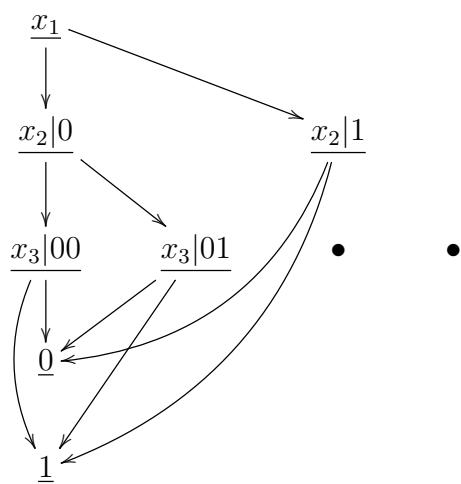


Figure 8.5: Image bnet for BDD of Fig.8.2.

Chapter 9

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

This chapter is mostly based on chapter 8 of Pearl's 1988 book Ref.[26]. See also Ref.[52] and references therein.

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

9.1 Chow-Liu Trees

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 $\underline{x}^n = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{n-1})$ 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

$$P(x^n) = P_{\underline{x}^n}(x^n). \quad (9.1)$$

Suppose we want to fit $P_{\underline{x}^n}$ by a tree bnet with nodes $\underline{t}^n = (\underline{t}_0, \underline{t}_1, \dots, \underline{t}_{n-1})$ 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

$$P_T(x^n) = P_{\underline{t}^n}(x^n). \quad (9.2)$$

Throughout this chapter, let $V = \{0, 1, \dots, n-1\}$, the set of vertices. Suppose μ is a function $\mu : V \rightarrow V$ such that $\mu(i) < i$. Let $T_\mu = \{\underline{t}_{\mu(i)} \rightarrow \underline{t}_i : i \in V - \{0\}\}$. 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:

$$P_T(x^n) = \prod_{i=0}^{n-1} P_T(x_i|x_{\mu(i)}) , \quad (9.3)$$

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

Claim 4 $D_{KL}(P \parallel P_T)$ is minimized over all probability distributions P_T that are expressible as Eq.(9.3) iff

$$P_T(x_i|x_{\mu(i)}) = P(x_i|x_{\mu(i)}) \quad (9.4)$$

for all i , and

$$\sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \quad (9.5)$$

is maximized over all μ .

proof:

$$D_{KL}(P \parallel P_T) = \sum_{x^n} P(x^n) \ln \frac{P(x^n)}{P_T(x^n)} \quad (9.6)$$

$$= - \sum_{x^n} \sum_i P(x^n) \ln P_T(x_i|x_{\mu(i)}) + \sum_i P(x^n) \ln P(x^n) \quad (9.7)$$

$$= - \sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \ln P_T(x_i|x_{\mu(i)}) - H(\underline{x}^n) \quad (9.8)$$

$$= - \sum_i \sum_{x_{\mu(i)}} P(x_{\mu(i)}) \left[\sum_{x_i} P(x_i|x_{\mu(i)}) \ln P_T(x_i|x_{\mu(i)}) \right] - H(\underline{x}^n) . \quad (9.9)$$

Now note that

$$\sum_{x_i} P(x_i|x_{\mu(i)}) \ln \frac{P(x_i|x_{\mu(i)})}{P_T(x_i|x_{\mu(i)})} \geq 0 \quad (9.10)$$

and this inequality becomes an equality iff

$$P(x_i|x_{\mu(i)}) = P_T(x_i|x_{\mu(i)}) . \quad (9.11)$$

Therefore

$$D_{KL}(P \parallel P_T) \geq - \underbrace{\sum_i \sum_{x_{\mu(i)}} P(x_{\mu(i)}) \left[\sum_{x_i} P(x_i|x_{\mu(i)}) \ln P(x_i|x_{\mu(i)}) \right]}_{=H(\underline{x}_i|x_{\mu(i)})=H(\underline{x}_i:x_{\mu(i)})-H(\underline{x}_i)} - H(\underline{x}^n) , \quad (9.12)$$

and this inequality becomes an equality iff Eq.(9.11) is satisfied.

Note from the last equation that

$$\operatorname{argmin}_{\mu} D_{KL}(P \parallel P_T) = \operatorname{argmax}_{\mu} \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) . \quad (9.13)$$

QED

Claim 5

$$\operatorname{argmin}_{\mu} H(\underline{x}^n) = \operatorname{argmax}_{\mu} \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \quad (9.14)$$

proof:

$$H(\underline{x}^n) = - \sum_{x^n} P(x^n) \sum_i \ln P(x_i | x_{\mu(i)}) \quad (9.15)$$

$$= - \sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \ln P(x_i | x_{\mu(i)}) \quad (9.16)$$

$$= - \sum_i \sum_{x_i, x_{\mu(i)}} P(x_i, x_{\mu(i)}) \left[\ln \frac{P(x_i | x_{\mu(i)})}{P(x_i)} + \ln P(x_i) \right] \quad (9.17)$$

$$= - \sum_i [H(\underline{x}_i : \underline{x}_{\mu(i)}) - H(\underline{x}_i)] \quad (9.18)$$

$$= \sum_i H(\underline{x}_i) - \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) \quad (9.19)$$

QED

The meaning of Claims 4 and 5 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

$$\text{score} = \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)}) . \quad (9.20)$$

(mutual information $H(a : b)$ measures correlation between a and 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 algorithm¹. (see Fig.9.1)

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.

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.9.2)

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 .

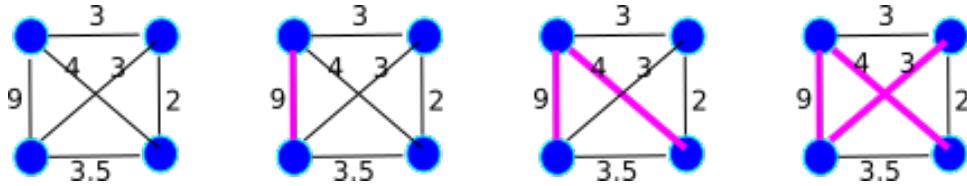


Figure 9.1: Example of finding MST (maximum spanning tree)

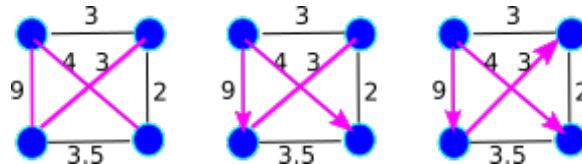


Figure 9.2: Example of giving directions to edges of spanning tree.

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} :

$$N_{nb}(\underline{a}) = \text{ number of neighbors of } \underline{a} \quad (9.21)$$

¹Kruskal'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[76] and references therein, or any other of numerous explanations of MST in the Internet.

$$\text{traffic}(\underline{a}) = \sum_{n \in nb(\underline{a})} H(\underline{a} : n) \quad (9.22)$$

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.

9.2 Tree Augmented Naive Bayes (TAN)

Recall from Chapter 38 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.9.3 and 9.4 for an example of a TAN bnet.

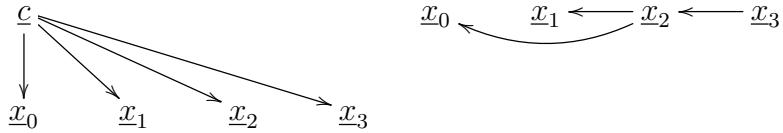


Figure 9.3: bnet for Naive Bayes with 4 feature nodes and another bnet for a tree made of the same feature nodes.

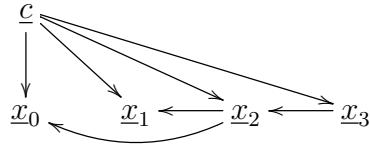


Figure 9.4: TAN bnet constructed by merging Naive Bayes bnet and tree bnet of Fig.9.3.

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

$$P_{TAN}(x^n, c) = P_{TAN}(c) \prod_{i=0}^{n-1} P_{TAN}(x_i | x_{\mu(i)}, c) . \quad (9.23)$$

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

Note that

Claim 6

$$\operatorname{argmin}_{\mu} H(\underline{x}^n, \underline{c}) = \operatorname{argmax}_{\mu} \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) \quad (9.24)$$

proof:

$$H(\underline{x}^n, \underline{c}) = - \sum_{x^n, c} P(x^n, c) \left[\ln P(c) + \sum_i \ln P(x_i | x_{\mu(i)}, c) \right] \quad (9.25)$$

$$= - \sum_{x^n, c} P(x^n, c) \left[\ln P(c) + \sum_i \ln \left(\frac{P(x_i, x_{\mu(i)} | c)}{P(x_i | c) P(x_{\mu(x_i)} | c)} P(x_i | c) \right) \right] \quad (9.26)$$

$$= \sum_i H(\underline{x}_i, \underline{c}) - \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) \quad (9.27)$$

QED

Following the same line of reasoning that we followed for Chow-Liu trees, we conclude that:

If $D_{KL}(P \| P_{TAN})$ 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

$$\text{score} = \sum_i H(\underline{x}_i : \underline{x}_{\mu(i)} | \underline{c}) \quad (9.28)$$

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

Calculate a Chow-Liu Tree for each $c \in S_c$. For each of those trees, create a TAN bnet, and calculate its score given by Eq.(9.28). Keep the TAN bnet with the largest score.

Chapter 10

Counterfactual Reasoning

This chapter is mostly based on Ref.[28], a 2019 review of causality by Pearl.

This chapter assumes that the reader has read Chapter 15 on do-calculus and Chapter 31 on LDEN (linear deterministic systems with external noise).

10.1 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 35 on message passing is about rung 1. Chapter 15 on do-calculus is about rung 2. This chapter is dedicated to rung 3.

10.2 Two kinds of intervention operators

In Chapter 15, 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 $\kappa_{\underline{x} \rightarrow \underline{a}}(x)$.

The 2 types of intervention operators are defined graphically in Fig.10.1.

- 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 $\kappa_{\underline{x} \rightarrow \underline{b}}(5)$ (called κ because it creates konstant nodes) operates on arrows unlike the ρ operator which operates on nodes. $\kappa_{\underline{x} \rightarrow \underline{b}}(5)$ deletes arrow $\underline{x} \rightarrow \underline{b}$ and creates a new root node \underline{x}' and a new arrow $\underline{x}' \rightarrow \underline{b}$. 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).

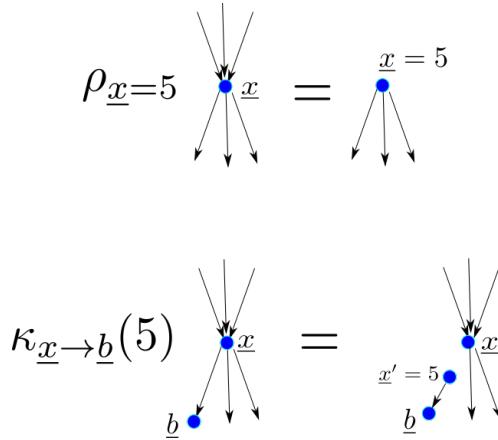


Figure 10.1: Action of “do” operator $\rho_{\underline{x}=5}$ on node \underline{x} and of “imagine” operator $\kappa_{\underline{x} \rightarrow \underline{b}}(5)$ on arrow $\underline{x} \rightarrow \underline{b}$.

10.3 Do operator 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.(31.21) of Chapter 31, are:

$$\underline{x} = A\underline{x} + \underline{u} . \quad (10.1)$$

Therefore,

$$\underline{x} = (1 - A)^{-1}\underline{u} \quad (10.2)$$

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

$$\underline{x}_i = x_i(\underline{u}) \quad (10.3)$$

If now we apply the operator $\rho_{\underline{a}=a}$ to the diagram described by the structural equations Eqs.10.1, we get the following new structural equations:

$$\underline{x}_i^* = \begin{cases} \sum_{j < i} A_{i|j} \underline{x}_j^* + \underline{u}_i & \text{if } \underline{x}_i \neq \underline{a} \\ \underline{a} & \text{if } \underline{x}_i = \underline{a} \end{cases} , \quad (10.4)$$

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

Eqs.(10.4) 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

$$(\pi_{\underline{a}})_{i,j} = \mathbb{1}(i = j, \underline{a} = \underline{x}_i) \quad (10.5)$$

and

$$(e_{\underline{a}})_i = \mathbb{1}(\underline{a} = \underline{x}_i) , \quad (10.6)$$

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

$$\pi_{!\underline{a}} = 1 - \pi_{\underline{a}} , \quad (10.7)$$

$$A^* = \pi_{!\underline{a}} A , \quad (10.8)$$

and

$$\underline{u}_{!\underline{a}} = \pi_{!\underline{a}} \underline{u} . \quad (10.9)$$

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

$$\underline{x}^* = A^* \underline{x}^* + \underline{u}_{!a} + ae_a . \quad (10.10)$$

Thus,

$$\underline{x}^* = (1 - A^*)^{-1}(\underline{u}_{!a} + ae_a) . \quad (10.11)$$

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

$$x_i^* = x_i^*(u_{!a}, a) . \quad (10.12)$$

For any bnet,

$$P(\underline{y} = y | \underline{x} = x) = P_G(\underline{y} = y | \underline{x} = x) \quad (10.13)$$

$$P(\underline{y} = y | \rho \underline{x} = x) = P_{\rho_{\underline{x}=x} G}(\underline{y} = y) \quad (10.14)$$

Claim 7 *For a non-linear DEN diagram,*

$$P(\underline{y} | \rho \underline{x} = x) = E [\delta[y, y(u_{!x}, x)]] . \quad (10.15)$$

proof:

$$P(\underline{y} = y | \rho \underline{x} = x) = P_{\rho_{\underline{x}=x} G}(\underline{y} = y) \quad (10.16)$$

$$= \sum_{u_{!x}} P(u_{!x}) P_{\rho_{\underline{x}=x} G}(\underline{y} = y | u_{!x}) \quad (10.17)$$

$$= \sum_{u_{!x}} P(u_{!x}) \delta[y, y(u_{!x}, x)] \quad (10.18)$$

$$= E_{u_{!x}} [\delta[y, y(u_{!x}, x)]] \quad (10.19)$$

$$= E[\delta[y, y(u_{!x}, x)]] \quad (10.20)$$

QED

Claim 8 *For a nonlinear DEN diagram,*

$$E[\underline{y} | \rho \underline{x} = x] = E[y(u_{!x}, x)] . \quad (10.21)$$

proof:

$$E[\underline{y}|\rho\underline{x} = x] = \sum_y y P(y = y|\rho\underline{x} = x) \quad (10.22)$$

$$= \sum_y y E[\delta[y, y(u_{!\underline{x}}, x)]] \quad (10.23)$$

$$= E[y(u_{!\underline{x}}, x)] \quad (10.24)$$

QED

For any bnet

$$P(y|\rho\underline{x} = x, z) = \frac{P(y, z|\rho\underline{x} = x)}{P(z|\rho\underline{x} = x)} = P_{\rho_{\underline{x}=x}G}(y|x, z) \quad (10.25)$$

For a nonlinear DEN diagram,

$$P(y, z|\rho\underline{x} = x) = \sum_{u_{!\underline{x}}} P(u_{!\underline{x}}) \delta[y, y(u_{!\underline{x}}, x)] \delta[z, z(u_{!\underline{x}}, x)] \quad (10.26)$$

$$P(z|\rho\underline{x} = x) = \sum_{u_{!\underline{x}}} P(u_{!\underline{x}}) \delta[z, z(u_{!\underline{x}}, x)]. \quad (10.27)$$

10.4 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.

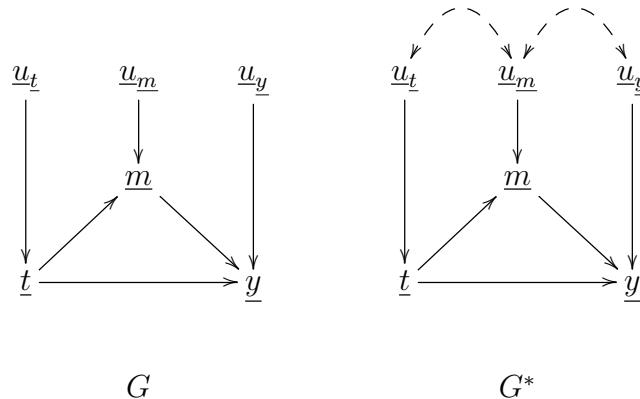


Figure 10.2: Graphs \$G\$ and \$G^*\$ are used to discuss mediation. In graph \$G\$, the exogenous variables are independent, whereas in graph \$G^*\$ they are not.

The term “mediation analysis” refers to the analysis of the DEN diagram G in Fig.10.2. 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:

$$\underline{t} = \underline{u}_t \quad (10.28a)$$

$$\underline{m} = f_{\underline{m}}(\underline{t}, \underline{u}_m) \quad (10.28b)$$

$$\underline{y} = f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_y) . \quad (10.28c)$$

Thus,

$$\underline{y} = f_{\underline{y}}(\underline{u}_t, f_{\underline{m}}(\underline{u}_t, \underline{u}_m), \underline{u}_y) . \quad (10.29)$$

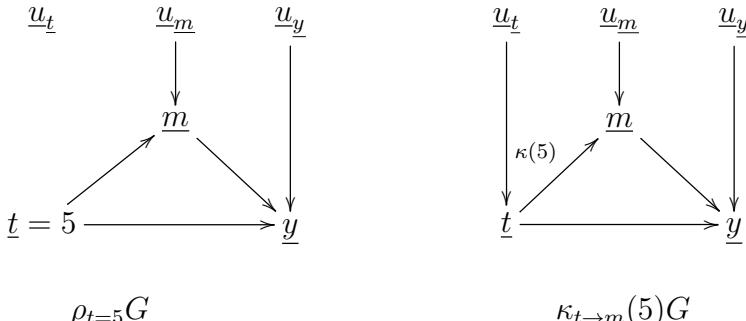


Figure 10.3: Graph G of Fig.10.2 after applying do operator $\rho_{\underline{t}=5}$ and imagine operator $\kappa_{\underline{t} \rightarrow \underline{m}}(5)$.

If we apply $\rho_{\underline{t}=5}G$ to Eqs.(10.28), we get

$$\underline{t} = 5 \quad (10.30a)$$

$$\underline{m} = f_{\underline{m}}(\underline{t}, \underline{u}_m) \quad (10.30b)$$

$$\underline{y} = f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_y) . \quad (10.30c)$$

Eqs.10.30 are represented graphically in Fig.10.3. We will often denote the random variable \underline{y} in Eqs.(10.30) by the more explicit symbol $\underline{y}_{\rho_{\underline{t}=5}G}$. Pearl often refers to this \underline{y} by the less explicit symbol Y_5 or $Y_5(u)$ where $Y = \underline{y}$ and $u = (\underline{u}_m, \underline{u}_y) = u_{!t}$.

If we apply $\kappa_{\underline{t} \rightarrow \underline{m}}(5)G$ to Eqs.(10.28), we get

$$\underline{t} = \underline{u}_t \quad (10.31a)$$

$$\underline{m} = f_{\underline{m}}(5, \underline{u}_m) \quad (10.31b)$$

$$\underline{y} = f_{\underline{y}}(\underline{t}, \underline{m}, \underline{u}_y) . \quad (10.31c)$$

Eqs.10.31 are represented graphically in Fig.10.3. We will often denote the random variable \underline{y} in Eqs.(10.31) by the more explicit symbol $\underline{y}_{\kappa_{\underline{t} \rightarrow \underline{m}}(5)G}$. Pearl often refers to this \underline{y} by the less explicit symbol Y_5 or $Y_5(u)$ where $Y = \underline{y}$ and $u = (u_{\underline{t}}, u_{\underline{m}}, u_{\underline{y}})$.

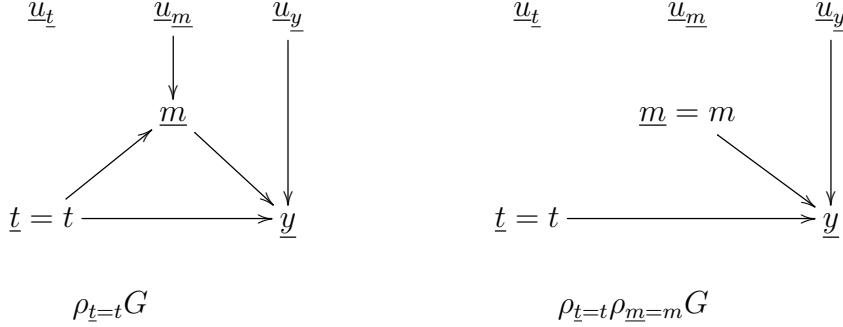


Figure 10.4: Graph G of Fig.10.2 after applying the do operators $\rho_{\underline{t}=t}$ and $\rho_{\underline{t}=t} \rho_{\underline{m}=m}$.

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

$$TE = E[\underline{y}_{\rho_{\underline{t}=1}G} - \underline{y}_{\rho_{\underline{t}=0}G}] \quad (10.32)$$

$$CDE(m) = E[\underline{y}_{\rho_{\underline{t}=1}\rho_{\underline{m}=m}G} - \underline{y}_{\rho_{\underline{t}=0}\rho_{\underline{m}=m}G}] \quad (10.33)$$

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.10.4.

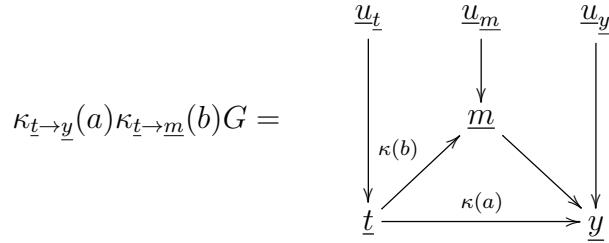


Figure 10.5: Graph G of Fig.10.2 after applying the imagine operator κ to arrows $\underline{t} \rightarrow \underline{m}$ and $\underline{t} \rightarrow \underline{y}$.

Let

$$\mathcal{Y}_a^b = E[\underline{y}_{\kappa_{\underline{t} \rightarrow \underline{y}}(a)\kappa_{\underline{t} \rightarrow \underline{m}}(b)G}] \quad (10.34)$$

Fig.10.5 shows the diagram $\kappa_{\underline{t} \rightarrow \underline{y}}(a)\kappa_{\underline{t} \rightarrow \underline{m}}(b)G$ used in the definition of \mathcal{Y}_a^b .

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

$$NDE = \mathcal{Y}_1^0 - \mathcal{Y}_0^0 \quad (10.35)$$

$$NIE(t) = \mathcal{Y}_t^1 - \mathcal{Y}_t^0. \quad (10.36)$$

Note that

$$NDE + NIE(1) = (\mathcal{Y}_1^0 - \mathcal{Y}_0^0) + (\mathcal{Y}_1^1 - \mathcal{Y}_1^0) \quad (10.37)$$

$$= \mathcal{Y}_1^1 - \mathcal{Y}_0^0 \quad (10.38)$$

$$= TE. \quad (10.39)$$

Chapter 11

Cross-Validation

This chapter is based on Ref.[53].

Cross-Validation (CV) is a method of calculating the “**out-of-training-set**” (**OOTS**) error for a classifier. What this means is that the classifier is trained on a training set, and its propensity to err is evaluated on a set different from the training set.

In **k -fold CV**, the most common CV method, and the only one we will discuss in this chapter, one partitions a dataset into k disjoint datasets of equal length. One uses $k - 1$ of those sub-datasets to train a model, and saves the last sub-dataset to validate the model just trained. One actually rotates which of the k sub-datasets is used for validation purposes, and calculates k validation errors \mathcal{E}_j for $j = 0, 1, \dots, k - 1$. Then one averages over the \mathcal{E}_j to obtain a final OOTS error \mathcal{E} .

CV strongly resembles Jackknife Resampling (JR) (see Chapter 27), but in JR the validation sub-dataset is never used for anything, whereas in CV, it is used for validation purposes, to calculate an OOTS error.

Next, we will explain k -fold CV more explicitly, using equations and a bnet.

Let $L = [0, 1, 2, \dots, nsam - 1]$ be a list of individuals (samples) in a population. In this chapter, we will use the notation $A^\sigma = A[\sigma]$ and $\vec{A} = [A^\sigma : \sigma \in L]$ for a list (vector, 1-D array) indexed by L . We will refer to $DS = (\vec{x}, \vec{y})$ where $x^\sigma \in S_x$, $y^\sigma \in S_y$, as a dataset. If L_j is a list (possibly with duplicate items) such that $set(L_j) \subset set(L)$, then define $DS_j = (\vec{x}, \vec{y})_{L_j} = ((x^\sigma)_{\sigma \in L_j}, (y^\sigma)_{\sigma \in L_j})$. We will refer to DS_j as the **restriction of (\vec{x}, \vec{y}) to L_j** .

Let $J = \{0, 1, 2, \dots, nj - 1\}$.

Define a **training list(TL), validation list(VL) pair** (TL, VL) to be a pair of lists such that $set(TL)$ and $set(VL)$ are disjoint subsets of $set(L)$. Let (TL_j, VL_j) for $j \in J$ be nj such TL-VL pairs.

Fig.11.1 shows the TL-VL pairs that are used when doing k -fold CV. In that figure, $k = nj = 4$. As you can see, in k -fold CV, one chooses $nj = k$ list pairs (TL_j, VL_j) such that all individuals $\sigma \in L$ appear exactly once, in either TL_j or VL_j , but not in both.

We will refer to a function $Y : S_x \rightarrow S_c$ as a classifier. It maps a vector of

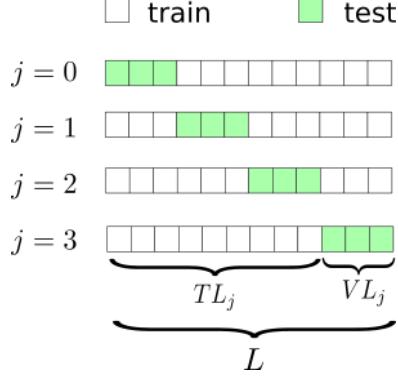


Figure 11.1: 4-fold CV with $|L| = 12$. For all j , $|VL_j| = 3$ and $|TL_j| = 9$. All individuals $\sigma \in L$ appear exactly once, in either TL_j or VL_j , but not in both.

features x to a class c . Let Y_j for $j \in J$ denote n_j classifiers.

If $Y_j : S_x \rightarrow S_c$ and S_c is a discrete set (“categorical”), then define the **OOTs error for the j th classifier** as:

$$\mathcal{E}_j = \frac{1}{|VL_j|} \sum_{\sigma \in VL_j} \mathbb{1}(y^\sigma \neq Y_j(x^\sigma)) . \quad (11.1a)$$

On the other hand, if $S_c = \mathbb{R}$, it makes more sense to define \mathcal{E}_j as a mean square error:

$$\mathcal{E}_j = \frac{1}{|VL_j|} \sum_{\sigma \in VL_j} (y^\sigma - Y_j(x^\sigma))^2 . \quad (11.1b)$$

Finally, define the **final OOTs error** as

$$\mathcal{E} = \frac{1}{|J|} \sum_{j \in J} \mathcal{E}_j . \quad (11.2)$$

Fig.11.2 gives a bnet that represents the CV algorithm. The TPMs, printed in blue, for the bnet of Fig.11.2, are as follows:

$$P((\vec{x}, \vec{y})_{TL_j} | (\vec{x}, \vec{y})) = \mathbb{1}(\text{ } (\vec{x}, \vec{y})_{TL_j} \text{ = restriction of } (\vec{x}, \vec{y}) \text{ to } TL_j.) \quad (11.3)$$

$$P((\vec{x}, \vec{y})_{VL_j} | (\vec{x}, \vec{y})) = \mathbb{1}(\text{ } (\vec{x}, \vec{y})_{VL_j} \text{ = restriction of } (\vec{x}, \vec{y}) \text{ to } VL_j.) \quad (11.4)$$

$$P(Y_j | (\vec{x}, \vec{y})_{TL_j}) = \mathbb{1}(\text{ } Y_j \text{ = classifier trained with } (\vec{x}, \vec{y})_{TL_j}) \quad (11.5)$$

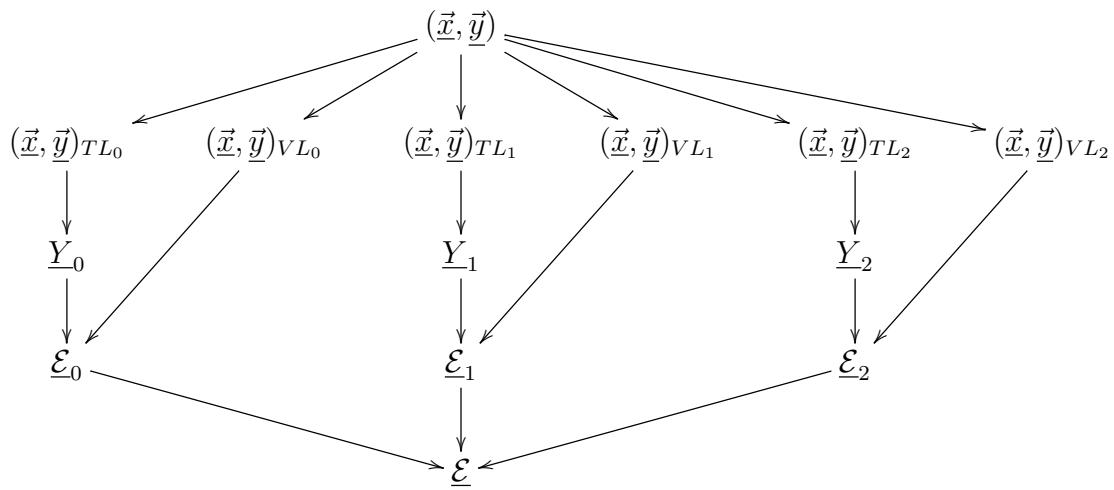


Figure 11.2: Bnet for 3-fold CV.

$$P(\mathcal{E}_j | Y_j, (\vec{x}, \vec{y})_{VL_j}) = \mathbb{1}(\mathcal{E}_j = \text{defined by Eqs.(11.1).}) \quad (11.6)$$

$$P(\mathcal{E} | (\mathcal{E}_j)_{j \in J}) = \mathbb{1}(\mathcal{E} = \text{defined by Eq.(11.2).}) \quad (11.7)$$

Chapter 12

Decision Trees

This chapter is based mainly on Ref.[37].

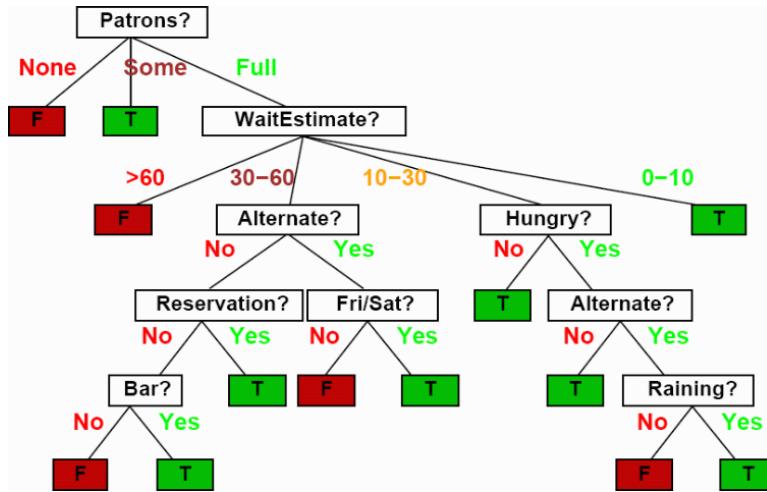


Figure 12.1: Example of dtree taken from Ref.[37]

Fig.12.1 shows a typical decision tree (dtree). This example was taken from Ref.[37], where it is analyzed in detail. As you can see, a dtree contains two main types of nodes: the non-leaf nodes, and the leaf nodes. The **non-leaf nodes** pose **questions**. In general, the **answers**¹ to those questions can be multiple choices with two or more choices. For each of those choices, a tree branch labeled by the choice comes down from the question node. The **leaf nodes** represent endpoints, goals, final conclusions, etc. Dtrees can be viewed as classifiers. They take in a large amount of information about a population and compress that information to just a few classes. If S_c is the set of distinct leaf node labels, then we call each $c \in S_c$ a **class of the classifier**. In the case of Fig.12.1, $S_c = \{False, True\}$.

Dtrees can be used with probabilities attached to each node, or without probabilities (as a plain undirected graph(UG)). This is analogous to bnets, which can

¹The **question-answer pairs** in dtrees are often also referred to as **attribute-value pairs**.

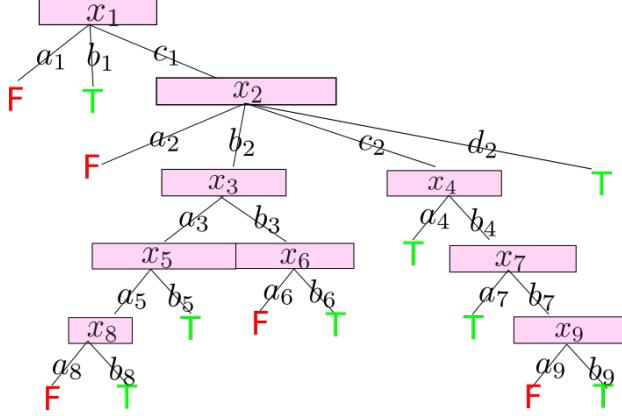


Figure 12.2: Fig.12.1 with abridged labels.

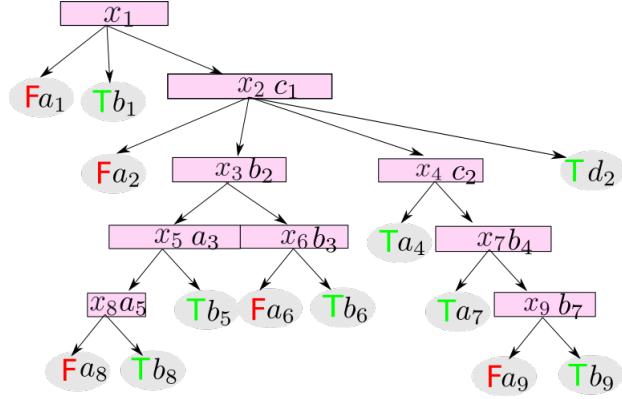


Figure 12.3: Fig.12.2 converted to a bnet.

be used with probabilities attached to each node (as DAGs with TPMs specified for each node) or without probabilities (as plain DAGs). Dtrees differ from bnets in that their tree branches are labelled, whereas bnet arrows aren't labelled. Also, whereas the nodes of a bnet carry a matrix of probabilities (the TPM), the nodes of a dtree carry just a column vector of probabilities which represents a single probability distribution. Henceforth, we will refer to the column vector of probabilities carried by each node of a dtree as its **Transition Probability Vector (TPV)**. Without the TPVs, a dtree can be used as a deterministic classifier, to classify inputs. With the TPVs, it can be used as a probabilistic sampler (to generate random samples.)

12.1 Transforming a dtree into a bnet

A trivial observation that is seldom made in the dtree pedagogical literature is that every dtree maps into a special bnet, let's call it its “image” bnet, in a very natural way. We use the dtree of Fig.12.1 as an example to show how to do this. As a first

$P(x a)$	$a = a_0$	$a \in S_{\underline{a}}^- - \{a_0\}$	$null$
0	p_0	0	0
1	p_1	0	0
\vdots	\vdots	0	0
$N_{\underline{x}}^- - 1$	$p_{N_{\underline{x}}^- - 1}$	0	0
$null$	0	1	1

Table 12.1: TPM of a node of a dtree image bnet.

step, we go from Fig.12.1 to Fig.12.2 by replacing all the labels of the nodes and of the branches of the dtree by abridged symbols. Next, we go from Fig.12.2 to Fig.12.3, by replacing all tree branches by arrows pointing down, and by moving the tree branch labels down so that they become a suffix to the question that the tree branch leads to. At this point, we have created Fig.12.3, which constitutes the DAG of the image bnet. It remains for us to define a TPM for each node of this DAG.

Table 12.1 gives the TPM $P(x|a)$ for a node \underline{x} with single parent \underline{a} of a dtree image bnet. Say node \underline{x} has a set $S_{\underline{x}}^-$ of possible tree branches coming out of it. Let $N_{\underline{x}}^- = |S_{\underline{x}}^-|$. Let $S_{\underline{x}} = S_{\underline{x}}^- \cup \{null\}$ and $N_{\underline{x}} = |S_{\underline{x}}| = N_{\underline{x}}^- + 1$. Define $S_{\underline{a}}^-, N_{\underline{a}}^-, S_{\underline{a}}$ and $N_{\underline{a}}$ analogously for node \underline{a} . In Table 12.1, $S_{\underline{x}}^- = \{0, 1, \dots, N_{\underline{x}}^- - 1\}$ and a_0 is the value of node \underline{a} which labels the tree branch connecting nodes \underline{a} and \underline{x} . $\vec{p} = (p_0, p_1, \dots, p_{N_{\underline{x}}^- - 1})$ is a probability distribution associated with node \underline{x} , its TPV. TPVs can be learned from a dataset following the dtree Structure Learning (SL) algorithm discussed in Section 12.2.

Table 12.1 also applies when node \underline{x} is a leaf node, except that for leaf nodes, \vec{p} is one hot (i.e., all components are zero except for one component which is 1). Also, all leaf nodes \underline{x} have the same $S_{\underline{x}}^-$, namely S_c .

Adding a null state to the set of states (SOS) of each node of the image bnet is necessary because, once $null$ is added to the SOS of any node, it must be added to the SOS of all descendant nodes. $null$ must be added to the SOS of the children of the root node to take care of the situations when those first children don't receive the state they were expecting from their parent, i.e., the root node.

When drawing dtrees, some people put info like explanations and probabilities on the branches of the dtree. That info can all be preserved in the TPM and the node names and node state names of the image bnet nodes. One can also place info inside tool tips attached to the node name and node state names. Often, the pedagogical 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 naive Bayes (NB) bnet (see Chapter 38) consists of a single “class node” with states S_c that fans out with arrows pointing to the “feature nodes”. If each leaf node of a NB bnet fans out into a set of new leaf nodes, and those new leaf nodes also fan out and so on, we get a generalized NB bnet. Let's call this type of tree bnet an NB^* bnet. An NB^* bnet has the same graph structure as the image bnet of a dtree,

but it's more general, because its TPMs are more general. Each TPM of a NB^* bnet can have several non-trivial columns instead of just one $\text{TPV} = \vec{p}$.

12.2 Structure Learning for Dtrees

Let

$J_0 = \{0, 1, \dots, nj - 1\}$
 $\Sigma = \{0, 1, 2, \dots, nsam - 1\}$
 $DS = \{(\sigma, x^\sigma, c^\sigma) : \sigma \in \Sigma\}$ be a dataset
 $\sigma \in \Sigma$ be an individual (a sample) from a population,
 $x^\sigma \in S_{\underline{x}}$ be the **feature (attributes, questions) vector**. $S_{\underline{x}} = S_{x_0} \times S_{x_1} \times \dots \times S_{x_{nj-1}}$, $x = (x_0, x_1, \dots, x_{nj-1}) \in S_{\underline{x}}$, $x_j \in S_{x_j}$
 $c^\sigma \in S_{\underline{c}}$ be a **classification class**
 We will assume S_x and S_c are finite sets.

Building a **classifier** Y (curve fit) for a dtree means finding a deterministic function $Y : S_{\underline{x}} \rightarrow S_{\underline{c}}$ such that $c^\sigma \approx Y(x^\sigma)$ for all $\sigma \in \Sigma$. If we divide the population Σ into two large disjoint sets, a **training set** Σ_{train} and a **validation set** Σ_{vali} , and if $c^\sigma \approx Y(x^\sigma)$ very closely for $\sigma \in \Sigma_{train}$ but fits poorly for $\sigma \in \Sigma_{vali}$, then we say the classifier Y suffers from **overfitting**. We can learn the structure and TPVs of a dtree from a dataset DS , by using the dtree **Structure Learning (SL)** algorithm that we will discuss in detail later. However, that algorithm is prone to produce a classifier Y that overfits. Two techniques commonly used to reduce the effects of overfitting are **pruning** and **Random Forest (RF)** (see Chapter 45). Pruning just means somehow removing nodes that are too specific. An RF is an ensemble of dtrees that one averages over. In this chapter, we will only deal with a single dtree, not an ensemble of them.

Below, we give the standard algorithm for SL of a dtree, in the form of pseudo-code. But first, we define two quantities, Information Gain and Gini, that are used in that pseudo-code.

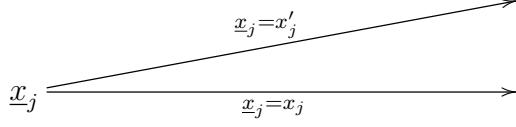
12.2.1 Information Gain, Gini

This section uses various Shannon Information Theory entropies. Our notation for those entropies is described in Chapter Notational Conventions and Preliminaries.

Call a **separation ability measure** (SAM) a measure used to decide, when constructing a dtree from a dataset, in what order to ask the questions about the feature vector x . The question order is decided by searching over all so far unused questions for the question with the largest SAM.²

Fig.12.4 defines some population numbers associated with the nodes of a dtree. From these population numbers, we can define the bnet in Fig.12.5. The TPMs,

²SAM is also called, somewhat confusingly, the splitting criterion and Gain.



$$\{N_j(c, x_j)\}_{c \in S_{\underline{c}}, x_j \in S_{\underline{x}_j}}$$

$$\sum_{c \in S_{\underline{c}}} N_j(c, x_j) = N_j(x_j)$$

$$\sum_{x_j \in S_{\underline{x}_j}} N_j(c, x_j) = N_j(c) \quad \sum_{c \in S_{\underline{c}}} N_j(c) = \sum_{x_j \in S_{\underline{x}_j}} N_j(x_j) = N_j$$

Figure 12.4: Some population numbers associated with the nodes of a dtree. $N_j(c, x_j)$ is the number of individuals σ in the population that reaches node j , belonging to class c and having $\underline{x}_j = x_j$.

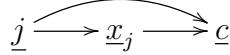


Figure 12.5: Bnet derived from population numbers in Fig.12.4

printed in blue, for the (non-root) nodes of this bnet, are as follows

$$P(c|x_j, j) = \frac{N_j(c, x_j)}{N_j(x_j)} \quad (12.1)$$

$$P(x_j|j) = \frac{N_j(x_j)}{N_j} \quad (12.2)$$

where $j \in J_0$ $x_j \in S_{\underline{x}_j}$, and $c \in S_{\underline{c}}$ is a class node.

One can define the following information theory quantities³ associated with the bnet Fig.12.5.

$$INFO_fin_j = - \sum_c P(c|j) \ln P(c|j) \quad (12.3)$$

$$= H(\underline{c}|j) \quad (12.4)$$

$$(12.5)$$

³ The average of $H(\underline{c} : b)$ over b is $H(\underline{c} : \underline{b}) = \sum_b P(b)H(\underline{c} : b)$. Likewise, the average of $H(\underline{c}|b)$ over b is $H(\underline{c}|b) = \sum_b P(b)H(\underline{c}|b)$. $H(\underline{c} : \underline{b})$ becomes $H(\underline{c} : b)$ and $H(\underline{c}|b)$ becomes $H(\underline{c}|b)$ when there is no b -prior (i.e., $P(b)$ is a delta function).

$$INFO_init_j = \sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) H(\underline{c}|x_j, j) \quad (12.6)$$

$$= H(\underline{c}|\underline{x}_j, j) \quad (12.7)$$

$$IG_j = INFO_fin_j - INFO_init_j \quad (12.8)$$

$$= H(\underline{c}|j) - H(\underline{c}|\underline{x}_j, j) \quad (12.9)$$

$$= H(\underline{c} : \underline{x}_j | j) \quad (12.10)$$

IG_j is called the **information gain for node \underline{x}_j** . Maximizing this mutual information produces a node \underline{x}_j that has a large correlation to a class c . If the goal is to reach a point where each leaf node is closely correlated to a different class, then maximizing the Information Gain of each new node is a greedy move towards that goal. Thus, Information Gain is a good SAM for dtree SL.

Note that if we approximate

$$\ln P(c|x_j) \approx \ln[1 + P(c|x_j) - 1] \quad (12.11)$$

$$\approx P(c|x_j) - 1 \quad (12.12)$$

in $H(\underline{c}|x_j)$, we get what is called the **Gini (or Gini Index) for node \underline{x}_k** :

$$H(\underline{c}|x_j) = - \sum_c P(c|x_j) \ln P(c|x_j) \quad (12.13)$$

$$\approx 1 - \sum_{c \in S_{\underline{c}}} P(c|x_j)^2 \stackrel{\text{def}}{=} Gini_{x_j} \quad (12.14)$$

$Gini_{x_j}$ is a fairly good polynomial approximation to $H(\underline{c}|x_j)$. It is computationally much less expensive than $H(\underline{c}|x_j)$, because it does not require computing a log.

We say a probability distribution $P_{\underline{x}}$, is **pure (i.e., deterministic)** if $P_{\underline{x}}(x) = \delta(x, x_0)$. $Gini_{x_j}$ and $H(\underline{c}|x_j)$ are both always non-negative. They both vanish iff $P(c|x_j)$ is pure. Thus, $Gini_{x_j}$ and $H(\underline{c}|x_j)$ are both good measures of **class impurity**.

The **average Gini of node \underline{x}_j** is defined as

$$AGini_j = \sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) Gini_{x_j} . \quad (12.15)$$

It measure the average impurity of the children of node \underline{x}_j .

In practice, the SL algorithm is done recursively. Each recursion step decides which feature x_j will be the root node of the current tree. For all “candidate” features (i.e., all \underline{x}_j that haven’t been used yet as tree nodes), one calculates IG_j , either exactly or approximately via Gini’s, using the following formula:

$$IG_j = \underbrace{H(\underline{c}|j)}_{\approx Gini_j} - \sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) \underbrace{H(\underline{c}|x_j)}_{\approx Gini_{x_j}} \quad (12.16)$$

One then chooses $j = \operatorname{argmax}_j IG_j$. This maximizes the $\underline{c} - \underline{x}_j$ correlation.

Alternatively, some software programs use the average Gini $AGini_j$ as their SAM. They choose as root node $j = \operatorname{argmin}_j AGini_j$. This minimizes the average impurity of the children of node j . Since IG_j differs from $AGini_j$ by $H(\underline{c}|j)$, maximizing IG_j and minimizing $AGini_j$ might lead to different results.

Example of calculation of $AGini_j$ and IG_j

Suppose we deduce from a dataset the numbers in the yellow cells in Table 12.2. These numbers are repeated in Fig.12.6. Then we can calculate the white cells as follows:

$j = \text{temp?}$	$x_j = \text{hot}$	$x_j = \text{med}$	$x_j = \text{cold}$
$c = \text{play}$	3	3	3
$c = \text{stay}$	1	2	2
$Gini$	0.375	0.48	0.48
$AGini = 0.45$			
$H(\underline{c} x_j)$	≈ 0.375	≈ 0.48	≈ 0.48
$IG \approx 0$			

Table 12.2: Evaluating AGini and IG for node *temp*.

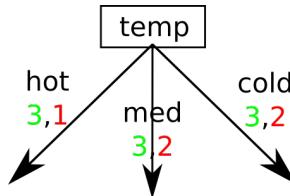


Figure 12.6: Tree stump corresponding to Table 12.2.

- $\text{Gini}(\text{hot}) = 1 - (3/4) - (1/4) = 0.375$
- $\text{Gini}(\text{med}) = 1 - (3/5)^2 - (2/5)^2 = 0.48$

- Gini(cold) = $1 - (3/5)^2 - (2/5)^2 = 0.48$
- AGini = $(4/14)(0.375) + (5/14)(0.48) + (5/14)(0.48) = 0.45$
- IG $\approx [1 - (6/9)^2 - (3/9)^2] - AGini = 0.44 - 0.45 \approx 0$

This gives the AGini and IG for just one candidate root node. We would have to calculate AGini (or IG) for all possible candidates and choose the candidate with the lowest AGini (or highest IG).

12.3 Information Gain Ratio

The Information Gain Ratio (IGR) is an alternative SAM.

$$IGR_j = \frac{IG_j}{H(\underline{x}_j|j)} \quad (12.17)$$

$$= \frac{H(\underline{c} : \underline{x}_j|j)}{H(\underline{x}_j|j)} \quad (12.18)$$

$$= \frac{H(\underline{x}_j|j) - H(\underline{x}_j|\underline{c}, j)}{H(\underline{x}_j|j)} \quad (12.19)$$

$$= 1 - \frac{H(\underline{x}_j|\underline{c}, j)}{H(\underline{x}_j|j)} \quad (12.20)$$

$$0 \leq IGR_j \leq 1$$

$$IGR_j = 0 \text{ iff } H(\underline{x}_j|\underline{c}, j) = H(\underline{x}_j|j).$$

12.3.1 Pseudo-code

Below, we give the standard algorithm for SL of a dtree, in the form of pseudo-code. The strategy employed by the algo is to assume an incoming population into the current root node, then determine the feature x_j that best separates that incoming population. The feature x_j is chosen so as to maximize IG_j . This process is repeated by nominating the end of each new branch to be the current root node. In essence, what we are doing is performing a top-down, greedy search through the space of possible dtrees.

The algo in the pseudo-code below is the essence of software programs called ID3 (Iterative Dichotomiser 3), C4.5/C5.0 (C4.5/C5.0 are successors to ID3) and CART (Classification and Regression Trees). ID3 (by Quinlan, 1986) and CART (by Breiman et al, 1984) are very similar, but were invented independently. As you can see, dtree SL is quite old. Many in the AI community consider it old fashioned compared to neural nets.

The pseudo-code below uses the majority function defined in Chapter Notational Conventions and Preliminaries.

Algorithm 1: Pseudo-code for learning a dtree from a dataset

Input : dataset $DS = \{(\sigma, x^\sigma, c^\sigma) : \sigma \in \Sigma\}$,
set of classification classes S_c ,
set of currently available node indices J , where $J \subset J_0$
Output: tree T ,
population numbers $\{(r, c, x_r, N_r(c, x_r)) : r \in J_0, c \in S_c, x_r \in S_{x_r}\}$
stored globally

```
 $J \leftarrow J_0$ 
Function learn_dtree( $DS, S_c, J$ ):
     $\Sigma \leftarrow$  set of all  $\sigma$  in  $DS$ 
    if  $\{c^\sigma : \sigma \in \Sigma\} = \{c\}$  then
         $T \leftarrow$  one node tree with leaf node label=  $c$ 
    else if  $J = \emptyset$  then
         $T \leftarrow$  one node tree with leaf node label= majority( $[c^\sigma : \sigma \in \Sigma]$ )
    else
         $r \leftarrow \underset{j \in J}{\text{argmax}} IG_j(DS)$ 
        from  $DS$ , calculate  $\{(r, c, x_r, N_r(c, x_r)) : c \in S_c, x_r \in S_{x_r}\}$  and store it
        globally
         $T \leftarrow$  a new decision tree with root node  $x_r$ 
        for  $v \in S_{x_r}$  do
            add a branch to tree  $T$  below  $x_r$  with label “ $x_r = v$ ”
             $DS_1 \leftarrow$  subset of  $DS$  with  $x_r = v$ 
            if  $DS_1 = \emptyset$  then
                below the new branch add a
                leaf node labeled = majority( $[c^\sigma : \sigma \in \Sigma]$ )
            else
                below the new branch add
                 $\text{subtree} = \text{learn\_dtree}(DS_1, S_c, J - \{j\})$ 
    return  $T$ 
```

Chapter 13

Difference-in-Differences

This chapter is based on Ref.[3].

The Difference-in-Differences (DID) method was first used by John Snow in an 1854 report that argued that cholera in London was being transmitted by sewage polluted water rather than, as others at the time believed, by air (in fetid vapors called miasmas). In general, one can apply DID to discover causal effects in historical data. By **historical data** (aka a **natural experiment**. See Ref.[81]) we mean data that is collected long after the treatment (rather than during it) and is thus not subject to active intervention by the experimenter.

This chapter assumes that the reader has read Chapter 43 on Potential Outcomes (PO). The DID method applies the basic single-time PO theory described in Chapter 43, to 2 well separated times in which different conditions prevail.

13.1 John Snow, DID and a cholera transmission pathway

Let

$$\begin{aligned} d &\in \{0, 1\} \\ t &\in \{t_0, t_1\}, t_0 < t_1 \\ y &= f(d, t) \in \mathbb{R}. \end{aligned}$$

Define

$$\Delta_t f(d, t) = f(d, t_1) - f(d, t_0), \quad (13.1)$$

$$\Delta_d f(d, t) = f(1, t) - f(0, t), \quad (13.2)$$

$$DID = \delta = \Delta_d \Delta_t f(d, t). \quad (13.3)$$

DID is illustrated in Fig.13.1.

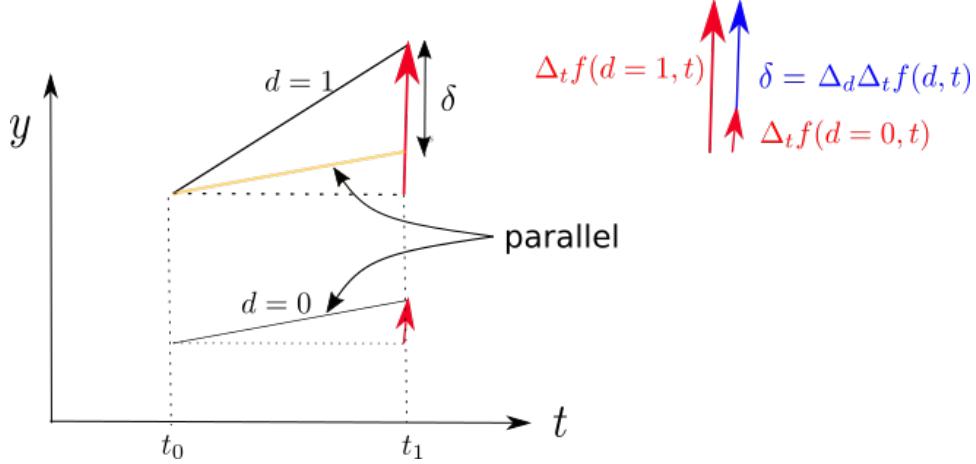


Figure 13.1: Pictorial representation of Difference-in-differences (DID) as a difference of two differences (i.e., a difference of two slopes).

A **time series** is any function of time for which the domain is a discrete set of times.

In DID, one calculates the slope, over the same time interval, of two time series. One of the time series ($d = 0$) is for the control (i.e., untreated) population and the other ($d = 1$) is for the treated population. Then the difference δ of those 2 slopes is taken. The idea is that if there is no causal difference between the 2 time series, then both time series will have the same slope, and δ will be zero.

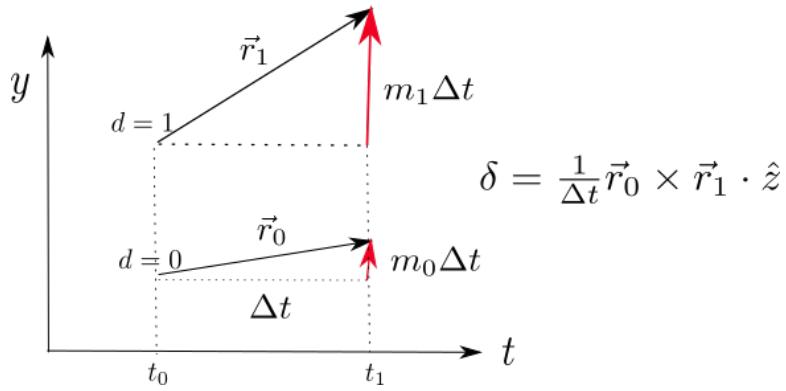


Figure 13.2: $DID = \delta$ expressed as a triple vector product.

Note that, as shown Fig.13.2, $DID = \delta$ can also be expressed as a triple vector product. Indeed, consider a space of points $(t, y, z) \in \mathbb{R}^3$ with an orthonormal basis $\hat{t}, \hat{y}, \hat{z}$. Let $\Delta t = t_1 - t_0$. Let m_d be the slope of the lines $d \in \{0, 1\} = \{ \text{untreated, treated} \}$. Let

$$\vec{r}_d = (\Delta t, m_d \Delta t, 0) \quad (13.4)$$

for $d \in \{0, 1\}$. Then

$$\frac{1}{\Delta t} \vec{r}_0 \times \vec{r}_1 \cdot \hat{z} = \frac{1}{\Delta t} \det \begin{bmatrix} \Delta t & m_0 \Delta t & 0 \\ \Delta t & m_1 \Delta t & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (13.5)$$

$$= (m_1 - m_0) \Delta t \quad (13.6)$$

$$= \delta \quad (13.7)$$

When \vec{r}_0 and \vec{r}_1 are parallel, $\vec{r}_0 \times \vec{r}_1 = 0$ so $\delta = 0$.

	$t = t_0$ (1849)	$t = t_1$ (1854)
$\tilde{d} = 1$ (town 1)	85 deaths, polluted DW	19 deaths, unpolluted DW
$\tilde{d} = 0$ (town 0)	135 deaths, polluted DW	147 deaths, polluted DW

Table 13.1: A condensation of the data collected by John Snow in 1854, to test the hypothesis that cholera in London was being spread by polluted drinking water (DW).

A condensation of the data collected by John Snow in 1854 is given in Table 13.1. From that data, we find that

$$\delta = \Delta_d \Delta_t f(d, t) = (19 - 85) - (147 - 135) = -66 - 12 = -78 \quad (13.8)$$

13.2 PO analysis

In this section, we show how to analyze the DID method using the formalism of PO theory.

We will speak of a treatment outcome $y_{t,g^\sigma}^\sigma(d^\sigma, \underline{x}^\sigma)$ for individual σ that depends, not just on the treatment dose $d^\sigma \in \{0, 1\}$ and the confounder state \underline{x}^σ , but also on a group parameter (i.e., which population or town) $g^\sigma \in \{0, 1\}$ and on a time parameter $t \in \{t_0, t_1\}$ (note t is independent of σ). Actually, we will assume $g^\sigma = d^\sigma$, so we will just speak of $y_t^\sigma(d^\sigma, \underline{x}^\sigma)$ with no explicit g^σ dependence. As usual for PO theory, we will consider expected values of y_t^σ :

$$E_{\sigma|\tilde{d},x}[y_t^\sigma(d)] = E_{y_t|\tilde{d},x}[y_t(d)] = \mathcal{Y}_{d|\tilde{d},x}(t) \quad (13.9)$$

To calculate these expected values, we need a “model” with probability distributions. In this case, the needed model and probability distributions are provided by the bnets depicted in Fig.13.3. The TPMs, printed in blue, for the bnet $G_{t,im}$ in Fig.13.3, are as follows. Note that the TPMs for the bnet $G_{t,im}$ are defined in terms of the TPMs for the bnet G_t .

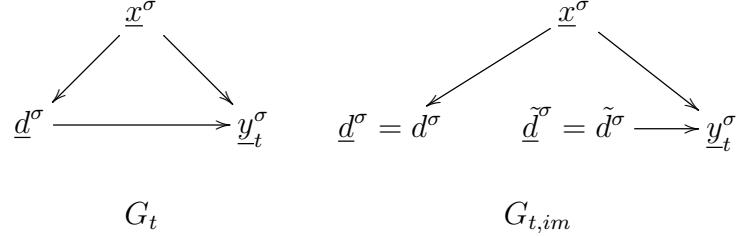


Figure 13.3: $t \in \{t_0, t_1\}$. Bnet $G_{t,im} = \kappa_{\underline{d}^{\sigma} \rightarrow \underline{y}_t^{\sigma}}(\tilde{d}^{\sigma})G_t$ is obtained by applying the imagine operator to arrow $\underline{d}^{\sigma} \rightarrow \underline{y}_t^{\sigma}$ of bnet G_t .

$$P(x^{\sigma}) = P_{\underline{x}}(x^{\sigma}) \quad (13.10)$$

$$P(d^{\sigma}|x^{\sigma}) = P_{\underline{d}|\underline{x}}(d^{\sigma}|x^{\sigma}) \quad (13.11)$$

$$P(y_t^{\sigma}|\tilde{d}^{\sigma}, x^{\sigma}) = P_{\underline{y}_t|\underline{d}, \underline{x}}(y_t^{\sigma}|\tilde{d}^{\sigma}, x^{\sigma}) \quad (13.12)$$

$$P((\tilde{d}^{\sigma})') = \delta((\tilde{d}^{\sigma})', \tilde{d}^{\sigma}) \quad (13.13)$$

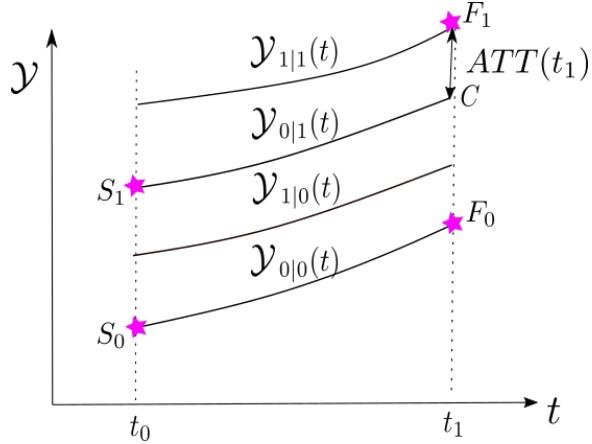


Figure 13.4: Four different time-dependent expected values $\mathcal{Y}_{d|\tilde{d}}(t)$ of y_t^{σ} for bnet $G_{t,im}$. The 4 magenta stars represent the 4 DID measurements.

Henceforth, for simplicity, we will omit the confounder state x from the indices of \mathcal{Y} ; i.e., we will write $\mathcal{Y}_{d|\tilde{d}}(t)$ instead of $\mathcal{Y}_{d|\tilde{d},x}(t)$. The fact that we will not explicitly

mention x does not mean that it doesn't exist or that it doesn't affect our analysis. John Snow does not seem to have considered any confounders in his cholera study, or else he tried to collect data restricted to a single stratum x . If there are confounders, they cannot be neglected. As discussed in Chapter 43 under the subject of strata-matching in PO, one must condition \mathcal{Y} on a single x stratum and, later on, one must average over all the possible x strata.

Let $\mathcal{MY}_{d|\tilde{d}}(t)$ denote the **measured** $\mathcal{Y}_{d|\tilde{d}}(t)$. We define this quantity as

$$\mathcal{MY}_{d|\tilde{d}}(t) = \mathcal{Y}_{d|\tilde{d}}(t) \left[\mathbb{1}(d=0, t=t_0) + \mathbb{1}(d=\tilde{d}, t=t_1) \right] \quad (13.14)$$

Now we claim that the DID δ calculated in the previous section for John Snow's data, can be expressed in PO formalism as follows:

$$\delta = \Delta_{\tilde{d}} \Delta_t \sum_d \mathcal{MY}_{d|\tilde{d}}(t) . \quad (13.15)$$

Fig.13.4 depicts the four functions $\mathcal{Y}_{d|\tilde{d}}(t)$ for t in the interval $[t_0, t_1]$ and for $d, \tilde{d} \in \{0, 1\}$. The \mathcal{Y} coordinates of the four magenta stars in Fig.13.4 can be calculated using bnet G_t .

Define the **parallel trends** (PT) by

$$PT = \Delta_{\tilde{d}} \Delta_t \mathcal{Y}_{0|\tilde{d}}(t) . \quad (13.16)$$

We will say the **parallel trends assumption (PTA)** holds if $PT = 0$.

Next we prove that the DID δ equals the sum of an ATT¹ and PT.

$$\delta = \Delta_{\tilde{d}} \Delta_t \sum_d \mathcal{MY}_{d|\tilde{d}}(t) \quad (13.17)$$

$$= \sum_d [\Delta_t \mathcal{MY}_{d|1}(t) - \Delta_t \mathcal{MY}_{d|0}(t)] \quad (13.18)$$

$$= \sum_d [\mathcal{MY}_{d|1}(t_1) - \mathcal{MY}_{d|1}(t_0)] - \sum_d [\Delta_t \mathcal{MY}_{d|0}(t_1) - \Delta_t \mathcal{MY}_{d|0}(t_0)] \quad (13.19)$$

$$= \mathcal{Y}_{1|1}(t_1) - \mathcal{Y}_{0|1}(t_0) - \{\mathcal{Y}_{0|0}(t_1) - \mathcal{Y}_{0|0}(t_0)\} \quad (13.20)$$

$$= \mathcal{Y}_{1|1}(t_1) - \mathcal{Y}_{0|1}(t_0) - \{\mathcal{Y}_{0|0}(t_1) - \mathcal{Y}_{0|0}(t_0)\} + \underbrace{\{\mathcal{Y}_{0|1}(t_1) - \mathcal{Y}_{0|1}(t_1)\}}_{\text{zero}} \quad (13.21)$$

$$= \underbrace{\mathcal{Y}_{1|1}(t_1) - \mathcal{Y}_{0|1}(t_1)}_{ATT(t_1)} - \mathcal{Y}_{0|1}(t_0) - \{\mathcal{Y}_{0|0}(t_1) - \mathcal{Y}_{0|0}(t_0)\} + \mathcal{Y}_{0|1}(t_1) \quad (13.22)$$

$$= ATT(t_1) - \Delta_t \mathcal{Y}_{0|0}(t) + \Delta_t \mathcal{Y}_{0|1}(t) \quad (13.23)$$

$$= ATT(t_1) + \underbrace{\Delta_{\tilde{d}} \Delta_t \mathcal{Y}_{0|\tilde{d}}(t)}_{\text{zero if PTA holds}} \quad (13.24)$$

¹ATT stands for the average treatment effect of the treated. ATT is defined in Chapter 43

13.3 Linear Regression

In this section, we show how to apply linear regression (LR) to the PO analysis of DID.

As before, let $y_t^\sigma(\tilde{d}^\sigma)$ be the treatment outcome for individual σ , who receives a treatment dose \tilde{d}^σ at times $t \in \{t_0, t_1\}$. $y_t^\sigma(\tilde{d}^\sigma)$ can be fitted as follows. Here ϵ^σ is the residual for individual σ and $b_0, m_0, b_1, m_1 \in \mathbb{R}$ are the fit parameters.

$$y_t^\sigma = [b_0 + m_0 t](1 - \tilde{d}^\sigma) + [b_1 + m_1 t]\tilde{d}^\sigma + \epsilon^\sigma . \quad (13.25)$$

Note that Eq.(13.25) yields a straight line in the $y_t^\sigma - t$ plane for $d^\sigma = 0$, and another straight line for $d^\sigma = 1$. We are using the standard symbols b to denote the y-intercept, and m to denote the slope of a straight line.

Taking the expected value of Eq.(13.25), we get

$$\mathcal{Y}_{d|\tilde{d}}(t) = [b_0 + m_0 t](1 - \tilde{d}) + [b_1 + m_1 t]\tilde{d} . \quad (13.26)$$

with $d = 0$ for $t = t_0$ and $d = \tilde{d}$ for $t = t_1$.

Let $\Delta t = t_1 - t_0$. Since $\Delta_t t = \Delta t$ and $\Delta_d d = 1$, one gets

$$\delta = \Delta_d \Delta_t \mathcal{Y}_{d|\tilde{d}}(t) = \Delta_d \Delta_t y_t^\sigma = (m_1 - m_0) \Delta t . \quad (13.27)$$

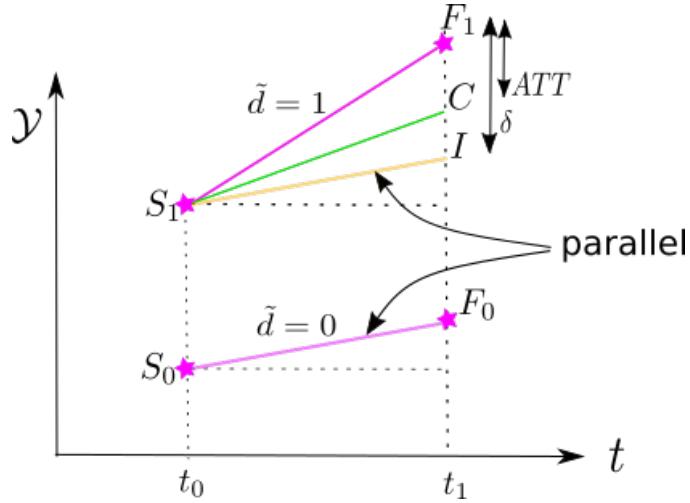


Figure 13.5: We use Linear Regression to fit a straight line between points S_0 and F_0 , and between points S_1 and F_1 . (S=starting, F=finishing). S_0, S_1, F_0, F_1 are the measurement points. Point I is an image of point F_0 . Point C is a counterfactual point.

	$t = t_0$	$t = t_1$
$\tilde{d} = 1$	$\mathcal{Y}(S_1) = \mathcal{Y}_{0 1}(t_0)$	$\mathcal{Y}(F_1) = \mathcal{Y}_{1 1}(t_1)$
$\tilde{d} = 0$	$\mathcal{Y}(S_0) = \mathcal{Y}_{0 0}(t_0)$	$\mathcal{Y}(F_0) = \mathcal{Y}_{0 0}(t_1)$

Table 13.2: \mathcal{Y} coordinates of points S_0, S_1, F_0, F_1 in Figs.13.4 and 13.5.

Figs.13.4 and 13.5 define points S_0, S_1, F_0, F_1, I, C . The \mathcal{Y} coordinates of points S_0, S_1, F_0, F_1 are given by Table 13.2. The \mathcal{Y} coordinates of points C, I are given by Eqs.13.28.

$$\mathcal{Y}(C) = \mathcal{Y}_{0|1}(t_1) \quad (13.28a)$$

$$\mathcal{Y}(I) = \mathcal{Y}(F_0) + [\mathcal{Y}(S_1) - \mathcal{Y}(S_0)] \quad (13.28b)$$

We can express *ATT* and the δ for DID in terms of the \mathcal{Y} of the points S_0, S_1, F_0, F_1, I, C . Indeed,

$$\delta = \mathcal{Y}(F_1) - \mathcal{Y}(I) \quad (13.29)$$

$$= \mathcal{Y}(F_1) - \mathcal{Y}(F_0) - [\mathcal{Y}(S_1) - \mathcal{Y}(S_0)] \quad (13.30)$$

$$ATT = \mathcal{Y}(F_1) - \mathcal{Y}(C) \quad (13.31)$$

Hence,

$$\delta = ATT \iff \mathcal{Y}(I) = \mathcal{Y}(C) \iff \text{PTA holds} \quad (13.32)$$

Chapter 14

Digital Circuits

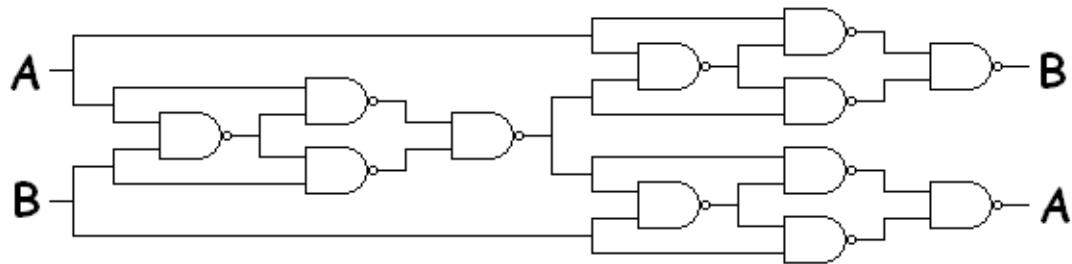


Figure 14.1: Typical digital circuit of NAND gates.

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

$$f : \{0, 1\}^{na} \rightarrow \{0, 1\}^{nx}. \quad (14.1)$$

Suppose

$$a^{na} = (a_i)_{i=0,1,\dots,na-1} \text{ where } a_i \in \{0, 1\},$$

$$x^{nx} = (x_i)_{i=0,1,\dots,nx-1} \text{ where } x_i \in \{0, 1\}.$$

f maps a^{na} into x^{nx} .

Digital circuit (dcircuit) = circuit of digital gates.

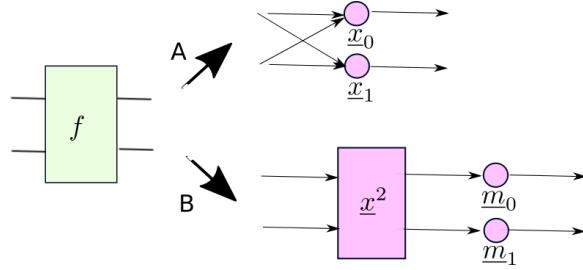


Figure 14.2: 2 options for mapping dcircuit node with multiple output ports into bnet.

14.1 Mapping any dcircuit to a bnet

14.1.1 Option A of Fig.14.2

1. Replace every dcircuit gate described by Eq.(14.1) by nx bnet nodes \underline{x}_i for $i = 0, 1, \dots, nx - 1$ such that

$$P(\underline{x}_i | a^{na}) = \delta(\underline{x}_i, f_i(a^{na})) \quad (14.2)$$

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

14.1.2 Option B of Fig.14.2

1. Replace every dcircuit gate described by Eq.(14.1) with one bnet node called x^{nx} and, if $nx > 0$, nx “marginalizer nodes” \underline{m}_i for $i = 0, 1, \dots, nx - 1$, such that

$$P(x^{nx} | a^{na}) = \delta(x^{nx}, f(a^{na})) , \quad (14.3)$$

and

$$P(\underline{m}_i | x^{nx}) = \delta(\underline{m}_i, x_i) . \quad (14.4)$$

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 15

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 Navigating the ocean of Judea Pearl's Books).

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.15.1 all mean the same thing.

A **confounder node** \underline{c} for nodes \underline{x} and \underline{y} is a root node with arrows pointing from it to both \underline{x} and \underline{y} . Thus, \underline{c} acts as a common cause of \underline{x} and \underline{y} . The node \underline{c} in Fig.15.1 is an **unobserved confounder node**.

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

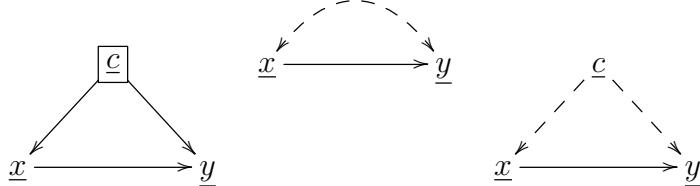


Figure 15.1: These 3 diagrams are equivalent. They mean that node \underline{c} is hidden. Node \underline{c} is implicit in the middle diagram.

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 converting \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.15.2.

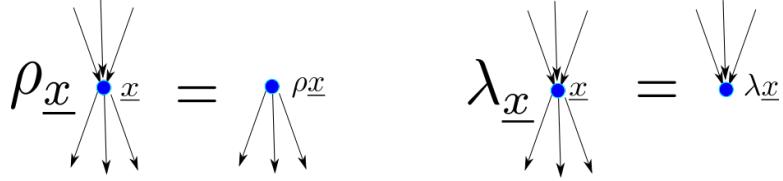


Figure 15.2: The operator $\rho_{\underline{x}}$ converts node \underline{x} into a root node $\rho\underline{x}$. The operator $\lambda_{\underline{x}}$ converts node \underline{x} into a leaf node $\lambda\underline{x}$.

If you don't know yet what we mean by a multi-node \underline{a} , see Chapter Definition of a Bayesian Network

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

$$\rho_{\underline{a}}.G = \left[\prod_j \rho_{\underline{a}_j} \right] G, \quad \lambda_{\underline{a}}.G = \left[\prod_j \lambda_{\underline{a}_j} \right] G. \quad (15.1)$$

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

$$P(X.) = \prod_j P(X_j | (X_k)_{k: \underline{X}_k \in \text{pa}(\underline{X}_j)}). \quad (15.2)$$

Define an operator ρ that acts as follows¹: Let $X. - a. = (X_k)_{k: \underline{X}_k \notin \underline{a}.}$

$$P(X. - a. | \rho \underline{a}. = a.) = \mathcal{N}(!(\underline{X} - a.)) \frac{P(X.)}{\prod_{j: \underline{X}_j \in \underline{a}.} P(X_j | (X_k)_{k: \underline{X}_k \in \text{pa}(\underline{X}_j)})} \quad (15.3)$$

$$= \mathcal{N}(!(\underline{X} - a.)) \prod_{j: \underline{X}_j \notin \underline{a}.} P(X_j | (X_k)_{k: \underline{X}_k \in \text{pa}(\underline{X}_j)}) \quad (15.4)$$

$$\neq P(X. - a. | \underline{a}. = a.). \quad (15.5)$$

Also,

$$P(X. - a., \rho \underline{a}. = a') = P(X. - a. | \rho \underline{a}. = a.) \delta(a', a.). \quad (15.6)$$

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

For instance, for the bnet

$$\underline{x} \longrightarrow \underline{y} \quad (15.7)$$

with

$$P(x, y) = P(y|x)P(x), \quad (15.8)$$

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

one has

$$P(y|\rho\underline{x} = x) = P(y|x) \quad (15.9)$$

and

$$P(x|\rho\underline{y} = y) = P(x) . \quad (15.10)$$

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

For the bnet



with

$$P(x, y, c) = P(y|x, c)P(x|c)P(c) , \quad (15.12)$$

one has

$$P(y, c|\rho\underline{x} = x) = P(y|x, c)P(c) . \quad (15.13)$$

Hence,

$$P(y|\rho\underline{x} = x) = \sum_c P(y|x, c)P(c) . \quad (15.14)$$

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

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

$$P(b.|\rho\underline{a.} = a.) = \sum_{X.-a.-b.} P(X.-a.|\rho\underline{a.} = a.) , \quad (15.15)$$

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

$$P(b.|\rho\underline{a.} = a., s.) = \frac{P(b., s.|\rho\underline{a.} = a.)}{P(s.|\rho\underline{a.} = a.)} . \quad (15.16)$$

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

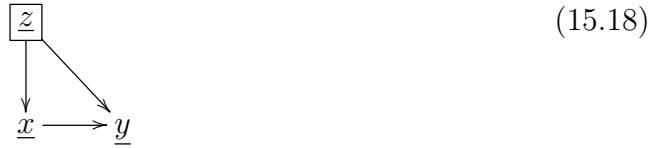
In $P(y|\rho\underline{x} = x)$, node \underline{x} is turned into a root node. This guarantees that there is no confounding node connecting \underline{x} and \underline{y} . Such confounding nodes are unwelcomed when calculating causal effects between the 2 variables \underline{x} and \underline{y} 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.

$P(b.|pa_{\cdot} = a_{\cdot}, s_{\cdot})$ is said to be **identifiable** (i.e., calculable) 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



but it is non-identifiable for the bnet



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

$$ACE = P(y = 1|\rho\underline{x} = 1) - P(y = 1|\rho\underline{x} = 0) \quad (15.19)$$

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

$$RD = P(y = 1|\underline{x} = 1) - P(y = 1|\underline{x} = 0). \quad (15.20)$$

15.1 Parent Adjustment

Suppose that $\underline{x}_{\cdot}, \underline{y}_{\cdot}, \underline{z}_{\cdot}$ 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 $P(x_{\cdot}, y_{\cdot}, z_{\cdot})$. Hence, all nodes of the bnet are observable. Furthermore, suppose $\underline{z}_{\cdot} = pa(\underline{x}_{\cdot})$. In other words, we are considering the bnet



Then

$$P(y_{\cdot}, z_{\cdot} | \rho\underline{x}_{\cdot} = x_{\cdot}) = P(y_{\cdot} | x_{\cdot}, z_{\cdot})P(z_{\cdot}) \quad (15.22)$$

so

$$P(y_{\cdot} | \rho_{\underline{x}} = x_{\cdot}) = \sum_{z_{\cdot}} P(y_{\cdot} | x_{\cdot}, z_{\cdot}) P(z_{\cdot}). \quad (15.23)$$

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

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}_{\cdot} 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}_{\cdot} , and $P(x_{\cdot}, y_{\cdot}, z_{\cdot})$ need not represent the whole bnet.

15.2 3 Rules of do-calculus

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

Recall from Chapter 16 on d-separation, that $(\underline{b}_{\cdot} \perp_G \underline{a}_{\cdot} | \underline{r}_{\cdot}, \underline{s}_{\cdot})$ means that we have established from the d-separation rules that all paths in G from \underline{a}_{\cdot} to \underline{b}_{\cdot} are blocked if we condition on $\underline{r}_{\cdot} \cup \underline{s}_{\cdot}$. Recall also that:

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

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

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

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.

15.3 Backdoor Adjustment

See Chapter 3 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.

For any two disjoint multinodes $\underline{x}.$ and $\underline{y}.$, we define a **backdoor path** from $\underline{x}.$ to $\underline{y}.$ as a path from $\underline{x}.$ and $\underline{y}.$ that starts with an arrow pointing into $\underline{x}.$,

Suppose that we have access to data that allows us to estimate a probability distribution $P(\underline{x}., \underline{y}., \underline{z}.)$. Hence, the variables $\underline{x}., \underline{y}., \underline{z}.$ are ALL the 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 backdoor paths from $\underline{x}.$ to $\underline{y}.$ are blocked by $\underline{z}..$
2. $\underline{z}.. \cap de(\underline{x}.) = \emptyset$.

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 9 *Backdoor Adjustment Theorem*

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

$$P(\underline{y}. | \rho \underline{x}. = \underline{x}.) = \sum_{\underline{z}..} P(\underline{y}. | \underline{x}., \underline{z}..) P(\underline{z}..) \quad (15.24)$$

$$\begin{aligned} &= \sum_{\underline{z}..} \begin{array}{c} \underline{z}.. = z.. \\ \searrow \\ \underline{x}.. = x.. \longrightarrow \underline{y}.. \end{array} \quad (15.25) \end{aligned}$$

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.



$$\begin{aligned}
& P(y|\rho\underline{x} = x) = \\
&= \sum_m P(y|\rho\underline{x} = x, z)P(z|\rho\underline{x} = x) \\
&\quad \text{by Probability Axioms} \\
&= \sum_P (y|x, z)P(z|\rho\underline{x} = x) \\
&\quad P(y|\rho\underline{x} = x, z) \rightarrow P(y|x, z) \\
&\quad \text{by Rule 2: If } (\underline{b} \perp \underline{a} | \underline{r}, \underline{s}) \text{ in } \lambda_{\underline{a}}\rho_{\underline{r}}G, \text{ then } P(b|\rho a_1 = a_1, \rho r_1 = r_1, s_1) = P(b|a_1, \rho r_1 = r_1, s_1). \\
&\quad \underline{y} \perp \underline{x} | \underline{z} \text{ in } \lambda_{\underline{x}}G \quad \begin{array}{ccc} \underline{z} & & \\ \downarrow & \searrow & \\ \underline{x} & & \underline{y} \end{array} \\
&= \sum_z P(y|x, z)P(z) \\
&\quad P(z|\rho\underline{x} = x) \rightarrow P(z) \\
&\quad \text{by Rule 3: If } (\underline{b} \perp \underline{a} | \underline{r}, \underline{s}) \text{ in } \rho_{\underline{a}-an(\underline{s})}\rho_{\underline{r}}G, \text{ then } P(b|\rho a_1 = a_1, \rho r_1 = r_1, s_1) = P(b|\rho r_1 = r_1, s_1). \\
&\quad \underline{z} \perp \underline{x} \text{ in } \rho_{\underline{x}}G \quad \begin{array}{ccc} \underline{z} & & \\ & \searrow & \\ \underline{x} & \longrightarrow & \underline{y} \end{array}
\end{aligned} \tag{15.27}$$

QED

Note that the backdoor adjustment formula can be written as

$$P(y_1|\rho\underline{x}_1 = x_1) = \sum_{z_1} P(y_1|x_1, z_1)P(z_1) \tag{15.28}$$

$$= \sum_{z_1} \frac{P(y_1, x_1, z_1)}{P(x_1|z_1)} \tag{15.29}$$

This assumes $P(x_1|z_1) \neq 0$ for all x_1, z_1 . This assumption is referred to as **positivity**, and is violated if $P(x_1|z_1) = \delta(x_1, x_1(z_1))$. $P(x_1|z_1)$ is called the **propensity score** of x_1 given z_1 . This equation does **inverse probability weighting**. One can approximate $P(x_1|z_1)$ in this equation to get an approximation to $P(y|\rho\underline{x} = x)$.

15.4 Front Door Adjustment

See Chapter 19 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 that we have access to data that allows us to estimate a probability distribution $P(x_1, m_1, y_1)$. Hence, the variables $\underline{x}_1, \underline{m}_1, \underline{y}_1$ are ALL the observed (i.e, not hidden). Then we say that the front-door \underline{m}_1 satisfies the **front-door adjustment criterion** relative to $(\underline{x}_1, \underline{y}_1)$ 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 (by \emptyset).
3. All backdoor paths from $\underline{m}.$ to $\underline{y}.$ are blocked by $\underline{x}..$

Claim 10 *Front-Door Adjustment Theorem*

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

$$P(y.| \rho \underline{x}. = x.) = \sum_{m.} \underbrace{\left[\sum_{x'} P(y.| x', m.) P(x'.) \right]}_{P(y.| \rho \underline{m}. = m.)} \underbrace{P(m.| x.)}_{P(m.| \rho \underline{x}. = x.)} \quad (15.30)$$

$$= \sum_{m., x'} \quad \begin{matrix} \underline{x} = x' \\ \searrow \\ \underline{x} = x. \longrightarrow \underline{m} = m. \longrightarrow \underline{y} \end{matrix} \quad (15.31)$$

proof: (See also Ref.[23] 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 c is hidden.



Continues in next page.

$$\begin{aligned}
& P(y|\rho\underline{x} = x) = \\
&= \sum_m P(y|\rho\underline{x} = x, m)P(m|\rho\underline{x} = x) \\
&\quad \text{by Probability Axioms} \\
&= \sum_m P(y|\rho\underline{x} = x, \rho\underline{m} = m)P(m|\rho\underline{x} = x) \\
&\quad P(y|\rho\underline{x} = x, m) \rightarrow P(y|\rho\underline{x} = x, \rho m = m) \\
&\quad \text{by Rule 2: If } (\underline{b}_. \perp \underline{a}_. | \underline{r}_., \underline{s}_.) \text{ in } \lambda_{\underline{a}}.\rho_{\underline{r}}.G, \text{ then } P(b.| \rho a_ = a_., \rho r_ = r_., s_.) = P(b.| a_., \rho r_ = r_., s_.). \\
&\quad \underline{y} \perp \underline{m} | \underline{x} \text{ in } \lambda_{\underline{m}}\rho_{\underline{x}}G \quad \boxed{C} \quad \begin{array}{ccc} & \searrow & \\ \underline{x} & \longrightarrow & \underline{m} & \underline{y} \end{array} \\
&= \sum_m P(y|\rho\underline{x} = x, \rho\underline{m} = m)P(m|x) \\
&\quad P(m|\rho\underline{x} = x) \rightarrow P(m|x) \\
&\quad \text{by Rule 2: If } (\underline{b}_. \perp \underline{a}_. | \underline{r}_., \underline{s}_.) \text{ in } \lambda_{\underline{a}}.\rho_{\underline{r}}.G, \text{ then } P(b.| \rho a_ = a_., \rho r_ = r_., s_.) = P(b.| a_., \rho r_ = r_., s_.). \\
&\quad \underline{m} \perp \underline{x} \text{ in } \lambda_{\underline{x}}G \quad \boxed{C} \quad \begin{array}{ccc} & \swarrow & \searrow & \\ & \underline{x} & \underline{m} & \longrightarrow & \underline{y} \end{array} \\
&= \sum_m P(y|\rho\underline{m} = m)P(m|x) \\
&\quad P(y|\rho\underline{x} = x, \rho\underline{m} = m) \rightarrow P(y|\rho\underline{m} = m) \\
&\quad \text{by Rule 3: If } (\underline{b}_. \perp \underline{a}_. | \underline{r}_., \underline{s}_.) \text{ in } \rho_{\underline{a}-an(\underline{s})}.\rho_{\underline{r}}.G, \text{ then } P(b.| \rho a_ = a_., \rho r_ = r_., s_.) = P(b.| \rho r_ = r_., s_.). \\
&\quad \underline{y} \perp \underline{x} | \underline{m} \text{ in } \rho_{\underline{x}}\rho_{\underline{m}}G \quad \boxed{C} \quad \begin{array}{ccc} & & \\ \underline{x} & \quad & \underline{m} & \longrightarrow & \underline{y} \end{array} \\
&= \sum_{x'} \sum_m P(y|\rho\underline{m} = m, x')P(x'|\rho\underline{m} = m)P(m|x) \\
&\quad \text{by Probability Axioms} \\
&= \sum_{x'} \sum_m P(y|m, x')P(x'|\rho\underline{m} = m)P(m|x) \\
&\quad P(y|\rho\underline{m} = m, x') \rightarrow P(y|m, x') \\
&\quad \text{by Rule 2: If } (\underline{b}_. \perp \underline{a}_. | \underline{r}_., \underline{s}_.) \text{ in } \lambda_{\underline{a}}.\rho_{\underline{r}}.G, \text{ then } P(b.| \rho a_ = a_., \rho r_ = r_., s_.) = P(b.| a_., \rho r_ = r_., s_.). \\
&\quad \underline{y} \perp \underline{m} | \underline{x} \text{ in } \lambda_{\underline{m}}G \quad \boxed{C} \quad \begin{array}{ccc} & \swarrow & \searrow & \\ & \underline{x} & \underline{m} & \longrightarrow & \underline{y} \end{array} \\
&= \sum_{x'} \sum_m P(y|m, x')P(x')P(m|x) \\
&\quad P(x'|\rho\underline{m} = m) \rightarrow P(x') \\
&\quad \text{by Rule 3: If } (\underline{b}_. \perp \underline{a}_. | \underline{r}_., \underline{s}_.) \text{ in } \rho_{\underline{a}-an(\underline{s})}.\rho_{\underline{r}}.G, \text{ then } P(b.| \rho a_ = a_., \rho r_ = r_., s_.) = P(b.| \rho r_ = r_., s_.). \\
&\quad \underline{x} \perp \underline{m} \text{ in } \rho_{\underline{m}}G \quad \boxed{C} \quad \begin{array}{ccc} & \swarrow & \searrow & \\ & \underline{x} & \underline{m} & \longrightarrow & \underline{y} \end{array}
\end{aligned}$$

(15.33)

QED

Chapter 16

D-Separation

Before reading this chapter, I recommend that you read Chapter Definition of a Bayesian Network.

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 $(\underline{c} \cup de(\underline{c})) \cap \underline{Z} = \emptyset$ (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.16.1 for some examples of paths that are blocked or not blocked by a multinode \underline{Z} .

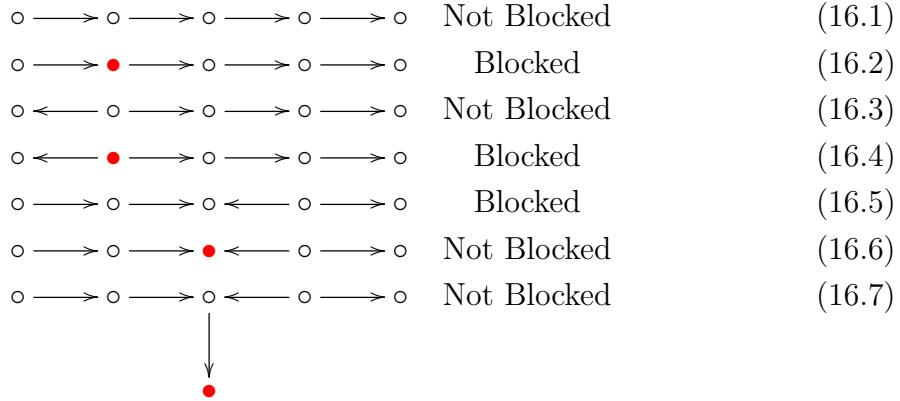


Figure 16.1: Examples of paths that are blocked or not blocked by a multinode $\underline{Z}.$ Nodes belonging to $\underline{Z}.$ are colored red.

Given 3 disjoint multinode $\underline{A}., \underline{B}., \underline{Z}.$ 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 $\underline{Z}.$ ¹

The minimal Markov blanket (see Chapter 32) of a node \underline{a} is the smallest multinode $\underline{Z}.$ such that $\underline{a} \perp_G \underline{b}|\underline{Z}.$ for all $\underline{b} \notin \underline{a} \cup \underline{Z}.$

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 11 (d-separation Theorem)

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

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

- $P(\underline{B}.|\underline{A}.,\underline{Z}.) = P(\underline{B}.|\underline{Z}.)$ for all $\underline{B}., \underline{A}., \underline{Z}..$
- $\underline{A}. \perp_P \underline{B}.|\underline{Z}.$

¹ $\underline{Z}.$ are the nodes we are “conditioning on”. Unmeasured (i.e., hidden, unobserved) nodes cannot be conditioned on, because that would entail measuring them.

- $H(\underline{A}_. : \underline{B}_. | \underline{Z}_.) = 0$ (see Chapter Notational Conventions and Preliminaries 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 Definition of a Bayesian Network.

Given a DAG G , define the following sets of d-separations:²

$$DS(G) = \{(\underline{A}_. \perp_G \underline{B}_. | \underline{Z}_.) : \underline{A}_., \underline{B}_., \underline{Z}_. \text{ are multinodes of } G\} . \quad (16.8)$$

$$DS_{min}(G) = \{(\underline{A}_. \perp_G nde(\underline{A}_.) | pa(\underline{A}_.)) : \underline{A}_. \text{ is a multinode of } G\} . \quad (16.9)$$

See Chapter 42 for an example where set $DS_{min}(G)$ is calculated for a particular DAG G .

Claim 12 *For all DAGs G , $DS(G) = DS_{min}(G)$.*

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

$$CI(P) = \{(\underline{A}_. \perp_P \underline{B}_. | \underline{Z}_.) : \underline{A}_., \underline{B}_., \underline{Z}_. \text{ are multinodes of } P\} , \quad (16.10)$$

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

$$\phi : DS_{min}(G) \rightarrow CI(P) \quad (16.11)$$

$$\phi : \underline{A}_. \perp_G nde(\underline{A}_.) | pa(\underline{A}_.) \mapsto \underline{A}_. \perp_P nde(\underline{A}_.) | pa(\underline{A}_.) \quad (16.12)$$

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

Claim 13 *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 42 for more info about d-separation equivalence.

² Note that $(\underline{A}_. \perp_G nde(\underline{A}_.) | pa(\underline{A}_.))$ and $(\underline{A}_. \perp_G nde(\underline{A}_.) - pa(\underline{A}_.) | pa(\underline{A}_.))$ are equivalent because $H(\underline{a} : \underline{b}, \underline{c} | \underline{c}) = H(\underline{a} : \underline{b} | \underline{c})$.

Chapter 17

Dynamic Bayesian Networks

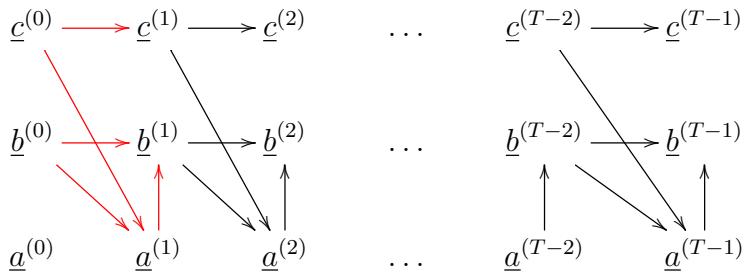


Figure 17.1: Example of a dynamic bnet. The pattern of red arrows is repeated $T - 1$ times.

A dynamic bnet is simply a time homogeneous Markov chain (see Chapter 34) 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.17.1 gives an example of a dynamic bnet. In Fig.17.1, we've drawn the 3 nodes of each time slice vertically, and labeled them with a superscript $^{(t)}$, where $t \in \{0, 1, \dots, T - 1\}$ is the time of the slice. To fully specify the dynamic bnet of Fig.17.1, we would also have to specify the TPMs

$$\begin{aligned} &P(\underline{c}^{(0)}), \\ &P(\underline{b}^{(0)}) \\ &P(\underline{c}^{(1)}|\underline{c}^{(0)}), \\ &P(\underline{b}^{(1)}|\underline{b}^{(0)}, \underline{a}^{(1)}) \\ &P(\underline{a}^{(1)}|\underline{b}^{(0)}). \end{aligned}$$

Dynamic bnets are very common in AI and Data Science. Kalman filters (Chapter 29), Hidden Markov Models (Chapter 23) and Recurrent Neural Networks (Chapter 46) 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. Fig.17.2 represents Fig.17.1

more compactly using feedback loops. Any bnet with feedback loops can be “unrolled” into a dynamic bnet.

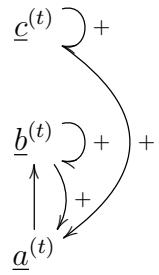


Figure 17.2: Dynamic bnet Fig.17.1 represented more compactly using feedback loops. Arrows labelled + point from nodes of the t time slice to nodes of the $t + 1$ time slice.

Chapter 18

Expectation Maximization

This chapter is based on Refs.[57] and [95].

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

$$P(\vec{x}|\theta) = \sum_{\vec{h}} P(\vec{x}, \vec{h}|\theta), \quad (18.1)$$

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

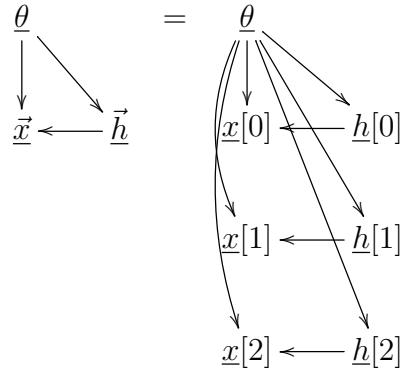


Figure 18.1: bnet for EM with $nsam = 3$.

The bnet for the EM algorithm is given by Fig.18.1 for $nsam = 3$. Later on in this chapter, we will give the node TPMs for this bnet for the special case in which $P(x[\sigma] | \theta)$ 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.18.1, 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.

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

$\vec{h} = (h[0], h[1], \dots, h[nsam - 1])$ $h[\sigma] \in S_{\underline{h}}$ 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 $P_{\underline{x}|\underline{h},\theta}$ and $P_{\underline{h}|\theta}$ such that

$$P(\vec{x}, \vec{h} | \theta) = \prod_{\sigma} [P_{\underline{x}|\underline{h},\theta}(x[\sigma] | h[\sigma], \theta) P_{\underline{h}|\theta}(h[\sigma] | \theta)] . \quad (18.2)$$

Definition of likelihood functions:

$$\underbrace{P(\vec{x} | \theta)}_{\mathcal{L}(\theta; \vec{x})} = \sum_{\vec{h}} \underbrace{P(\vec{x}, \vec{h} | \theta)}_{\mathcal{L}(\theta; \vec{x}, \vec{h})} \quad (18.3)$$

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

$$\theta^* = \operatorname{argmax}_{\theta} \mathcal{L}(\theta; \vec{x}) \quad (18.4)$$

18.1 The EM algorithm:

1. Expectation step:²

$$Q(\theta | \theta^{(t)}) = E_{\vec{h} | \vec{x}, \theta^{(t)}} \ln P(\vec{x}, \vec{h} | \theta) \quad (18.5)$$

2. Maximization step:

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)}) . \quad (18.6)$$

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

² Note that that the right hand side of Eq.(18.5) is expressible in the form $\sum_{\sigma} \sum_{h[\sigma]} f(x[\sigma], h[\sigma])$.

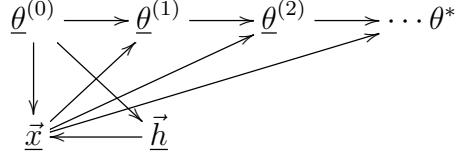


Figure 18.2: The EM algo generates a sequence of parameter estimates $(\underline{\theta}^{(t)})_{t=0,1,2,\dots}$ that converges to the optimum (i.e., best-fit) parameter $\underline{\theta}^*$.

Fig.18.2 portrays the recursive nature of the EM algo as a dynamical, recurrent bnet. For Fig.18.2, the TPMs, printed in blue, for the $\underline{\theta}^{(t)}$ nodes for $t = 1, 2, \dots$, are as follows:

$$P(\theta^{(t+1)} | \vec{x}, \theta^{(t)}) = \delta(\theta^{(t+1)}, \operatorname{argmax}_{\theta} Q(\theta | \theta^{(t)})) . \quad (18.7)$$

18.1.1 Motivation

$$Q(\theta | \theta^{(t)}) = E_{\vec{h} | \vec{x}, \theta^{(t)}} \ln P(\vec{x}, \vec{h} | \theta) \quad (18.8)$$

$$= E_{\vec{h} | \vec{x}, \theta^{(t)}} [\ln P(\vec{h} | \vec{x}, \theta) + \ln P(\vec{x} | \theta)] \quad (18.9)$$

$$= -D_{KL} \left(P(\vec{h} | \vec{x}, \theta^{(t)}) \| P(\vec{h} | \vec{x}, \theta) \right) - H[P(\vec{h} | \vec{x}, \theta^{(t)})] + \ln P(\vec{x} | \theta) \quad (18.10)$$

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

$$Q(\theta | \theta) = -H[P(\vec{h} | \vec{x}, \theta)] + \ln P(\vec{x} | \theta) . \quad (18.11)$$

Hence,

$$\partial_{\theta} Q(\theta | \theta) = - \sum_{\vec{h}} \partial_{\theta} P(\vec{h} | \vec{x}, \theta) + \partial_{\theta} \ln P(\vec{x} | \theta) \quad (18.12)$$

$$= \partial_{\theta} \ln P(\vec{x} | \theta) \quad (18.13)$$

So if $\theta^{(t)} \rightarrow \theta$ 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 $\lim_{t \rightarrow \infty} \theta^{(t)} = \theta^*$, see Wikipedia article Ref.[57] and references therein.

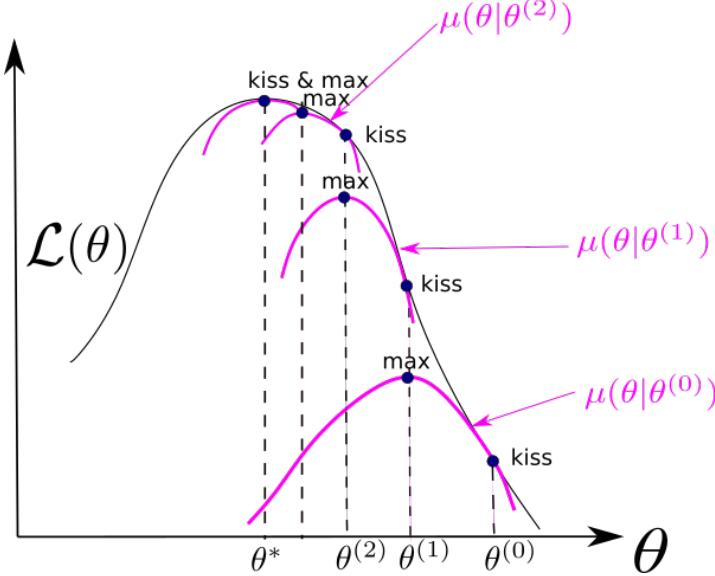


Figure 18.3: Function $\mu(\theta|\theta^{(t)})$ minorizes the function $L(\theta)$. Note that $\mu(\theta|\theta^{(t)})$ is always below $L(\theta)$. “max” indicates $\theta^{(t+1)} = \operatorname{argmax}_{\theta} \mu(\theta|\theta^{(t)})$. “kiss” indicates $\mu(\theta^{(t)}|\theta^{(t)}) = L(\theta^{(t)})$.

18.2 Minorize-Maximize (MM) algorithms

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

$$\mu(\theta|\theta^{(t)}) \leq L(\theta) , \quad (18.14)$$

and the “ $\mu = L$ property”

$$\mu(\theta^{(t)}|\theta^{(t)}) = L(\theta^{(t)}) . \quad (18.15)$$

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

$$\theta^{(t+1)} = \operatorname{argmax}_{\theta} \mu(\theta|\theta^{(t)}) . \quad (18.16)$$

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

$$L(\theta^{(t+1)}) \geq \mu(\theta^{(t+1)}|\theta^{(t)}) \geq \mu(\theta^{(t)}|\theta^{(t)}) = L(\theta^{(t)}) . \quad (18.17)$$

A **minorize-maximize (MM) algorithm** is any algo that specifies a minorizing function $\mu(\theta|\theta^{(t)})$ for a particular target function $L(\theta)$. 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

$$\mathcal{L}(\theta) = \ln P(\vec{x}|\theta) \quad (18.18)$$

and

$$\mu(\theta|\theta^{(t)}) = Q(\theta|\theta^{(t)}) + H(P(\underline{h}|\vec{x}, \theta^{(t)}), \quad (18.19)$$

then Eq.(18.10) 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.18.3.

18.3 Examples

18.3.1 Gaussian mixture

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

Let

$$\theta = [w_h, \mu_h, \Sigma_h]_{h \in S_h}, \quad (18.20)$$

where $[w_h]_{h \in S_h}$ 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.18.1, for the special case of a mixture of Gaussians, are as follows:

$$P(x[\sigma] | h[\sigma] | \theta) = \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]}) \quad (18.21)$$

$$P(h[\sigma] | \theta) = w_{h[\sigma]} \quad (18.22)$$

Note that

$$P(x[\sigma] | \theta) = \sum_h P(x[\sigma] | h[\sigma] = h, \theta) P(h[\sigma] = h | \theta) \quad (18.23)$$

$$= \sum_h w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h) \quad (18.24)$$

$$P(\vec{x}, \vec{h} | \theta) = \prod_{\sigma} [w_{h[\sigma]} \mathcal{N}_d(x[\sigma]; \mu_{h[\sigma]}, \Sigma_{h[\sigma]})] \quad (18.25)$$

$$= \prod_{\sigma} \prod_h [w_h \mathcal{N}_d(x[\sigma]; \mu_h, \Sigma_h)]^{\mathbb{1}(h=h[\sigma])} \quad (18.26)$$

Old Faithful: See Wikipedia Ref.[57] 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 $n_h = 2$. 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, \dots, 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.[67].

18.3.2 Blood Genotypes and Phenotypes

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

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

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

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

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

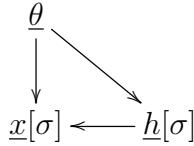


Figure 18.4: bnet for blood phenotypes $\underline{x}[\sigma]$ and genotypes $\underline{h}[\sigma]$.

For the bnet of Fig.18.4, the TPMs, printed in blue, are:

$$P(\underline{h}^{[\sigma]} | \theta) = \begin{array}{c|c} & \textcolor{blue}{AA} & \textcolor{blue}{p_A^2} \\ & \textcolor{blue}{AO} & \textcolor{blue}{2p_A p_O} \\ & \textcolor{blue}{BB} & \textcolor{blue}{p_B^2} \\ & \textcolor{blue}{BO} & \textcolor{blue}{2p_B p_O} \\ & \textcolor{blue}{OO} & \textcolor{blue}{p_O^2} \\ & \textcolor{blue}{AB} & \textcolor{blue}{2p_A p_B} \end{array}, \quad (18.27)$$

where $p_A + p_B + p_O = 1$.

$$P(\underline{x}^{[\sigma]} | \underline{h}^{[\sigma]}, \theta) = \begin{array}{c|cccccc} & \textcolor{blue}{AA} & \textcolor{blue}{AO} & \textcolor{blue}{BB} & \textcolor{blue}{BO} & \textcolor{blue}{OO} & \textcolor{blue}{AB} \\ \textcolor{blue}{A} & 1 & 1 & 0 & 0 & 0 & 0 \\ \textcolor{blue}{B} & 0 & 0 & 1 & 1 & 0 & 0 \\ \textcolor{blue}{O} & 0 & 0 & 0 & 0 & 1 & 0 \\ \textcolor{blue}{AB} & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \quad (18.28)$$

$$\theta = (p_A, p_B) \quad (18.29)$$

Multiplying the TPMs in Eqs.(18.27 and (18.28), we get

$$P(\underline{x}^{[\sigma]} | \theta) = \begin{array}{c|c} & \textcolor{blue}{A} & \textcolor{blue}{p_A^2 + 2p_A p_O (= \pi_A)} \\ & \textcolor{blue}{B} & \textcolor{blue}{p_B^2 + 2p_B p_O (= \pi_B)} \\ & \textcolor{blue}{O} & \textcolor{blue}{p_O^2 (= \pi_O)} \\ & \textcolor{blue}{AB} & \textcolor{blue}{2p_A p_B (= \pi_{AB})} \end{array} \quad (18.30)$$

Note that

$$P(\vec{x} | \theta) = \prod_{\sigma} P(x^{[\sigma]} | \theta) \quad (18.31)$$

$$= (\pi_A)^{N_A} (\pi_B)^{N_B} (\pi_O)^{N_O} (\pi_{AB})^{N_{AB}}, \quad (18.32)$$

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

$$\hat{\pi}_x = \frac{N_x}{N_+} \quad (18.33)$$

for $x \in S_{\underline{x}}$, where $N_+ = \sum_x N_x$. Eqs.(18.33) 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

$$Q(\theta|\theta^{(t)}) = \sum_{\vec{h}} P(\vec{h}|\theta^{(t)}) \ln P(\vec{x}, \vec{h}|\theta) \quad (18.34)$$

$$= \sum_{\vec{h}} \left[\prod_{\sigma} P(h^{[\sigma]}|\theta^{(t)}) \right] \ln \left[\prod_{\sigma} P(x^{[\sigma]}, h^{[\sigma]}|\theta) \right] \quad (18.35)$$

$$= \sum_{\sigma} \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(x^{[\sigma]}, h^{[\sigma]}|\theta) \quad (18.36)$$

$$= \sum_{\sigma} \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) [\ln P(x^{[\sigma]}|h^{[\sigma]}, \theta) + \ln P(h^{[\sigma]}|\theta)] \quad (18.37)$$

$$= nsam \sum_{h^{[\sigma]}} P(h^{[\sigma]}|\theta^{(t)}) \ln P(h^{[\sigma]}|\theta) . \quad (18.38)$$

18.3.3 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 36.

Chapter 19

Front-door Adjustment

The front-door (FD) adjustment theorem is proven in Chapter 15 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 15 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 16.

Suppose that 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 the 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 (by \emptyset).
3. All backdoor paths from $\underline{m}.$ to $\underline{y}.$ are blocked by $\underline{x}..$

Claim 14 *Front-Door Adjustment Theorem*

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

$$P(y.| \rho \underline{x}. = x.) = \sum_{m.} \underbrace{\left[\sum_{x'} P(y.| x', m.) P(x'.|) \right]}_{P(y.| \rho \underline{m}. = m.)} \underbrace{P(m.| x.)}_{P(m.| \rho \underline{x}. = x.)} \quad (19.1)$$

$$= \sum_{m., x'} \quad \begin{matrix} x. = x.' \\ \searrow \\ \underline{x}. = x. \longrightarrow \underline{m}. = m. \longrightarrow \underline{y} \end{matrix} \quad (19.2)$$

proof: See Chapter 15
QED

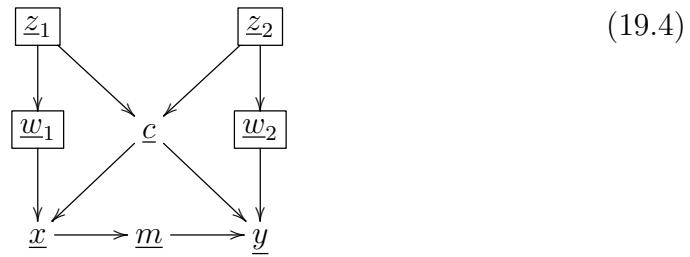
19.1 Examples

1.



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.



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 20

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 54).

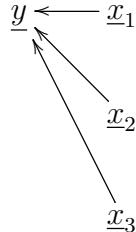


Figure 20.1: GLP node \underline{y} with 3 parent nodes $\underline{x}^3 = (\underline{x}_1, \underline{x}_2, \underline{x}_3)$.

Recall our notation for a Gaussian distribution:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad (20.1)$$

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

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

$$P(\underline{y}|\underline{x}^n) = \mathcal{N}(\underline{y}; \beta_0 + \beta^{nT} \underline{x}^n, \sigma^2) \quad (20.2)$$

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.20.1 shows a diagrammatic representation of a GLP node with 3 parents.

Note that as $\sigma \rightarrow 0$, 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 $\hat{\underline{y}}$ of a “true value” \underline{y} by

$$\hat{\underline{y}} = \beta_0 + \beta^{nT} \underline{x}^n \quad (20.3a)$$

and

$$\underline{y} = \hat{\underline{y}} + \underline{\epsilon} \quad (20.3b)$$

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

$$P(\epsilon) = \mathcal{N}(\epsilon; 0, \sigma^2) \quad (20.3c)$$

and

$$\langle \underline{x}^n, \underline{\epsilon} \rangle = 0 . \quad (20.3d)$$

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

Claim 15 *The parameters of a GLP node can be expressed in terms of 2-node covariances. Specifically,*

$$\beta^n = \langle \underline{x}^n, \underline{x}^{nT} \rangle^{-1} \langle \underline{y}, \underline{x}^n \rangle \quad (20.4)$$

$$\beta_0 = \langle \underline{y} \rangle - \beta^{nT} \langle \underline{x}^n \rangle \quad (20.5)$$

$$\sigma^2 = \langle \underline{y}, \underline{y} \rangle - \beta^{nT} \langle \underline{x}^n, \underline{y} \rangle \quad (20.6)$$

proof:

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

$$\langle \underline{y}, \underline{x}^{nT} \rangle = \beta^{nT} \langle \underline{x}^n, \underline{x}^{nT} \rangle \quad (20.7)$$

$$\langle \underline{y}, \underline{x}^n \rangle = \langle \underline{x}^n, \underline{x}^{nT} \rangle \beta^n \quad (20.8)$$

$$\beta^n = \langle \underline{x}^n, \underline{x}^{nT} \rangle^{-1} \langle \underline{y}, \underline{x}^n \rangle \quad (20.9)$$

$$\langle \underline{y} \rangle = \beta_0 + \beta^{nT} \langle \underline{x}^n \rangle \quad (20.10)$$

$$\langle \underline{y}, \underline{y} \rangle = \langle \beta_0 + \beta^{nT} \underline{x}^n + \underline{\epsilon}, \underline{y} \rangle \quad (20.11)$$

$$= \beta^{nT} \langle \underline{x}^n, \underline{y} \rangle + \sigma^2 \quad (20.12)$$

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

$$P(y|(x^n)_d, d) = \mathcal{N}(y; (\beta_0)_d + (\beta^{nT})_d(x^n)_d, \sigma_d^2) \quad (20.13)$$

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 sub-intervals and assign a discrete label to each subinterval.

- $D \leftarrow D$

Chapter 21

Generative Adversarial Networks (GANs)

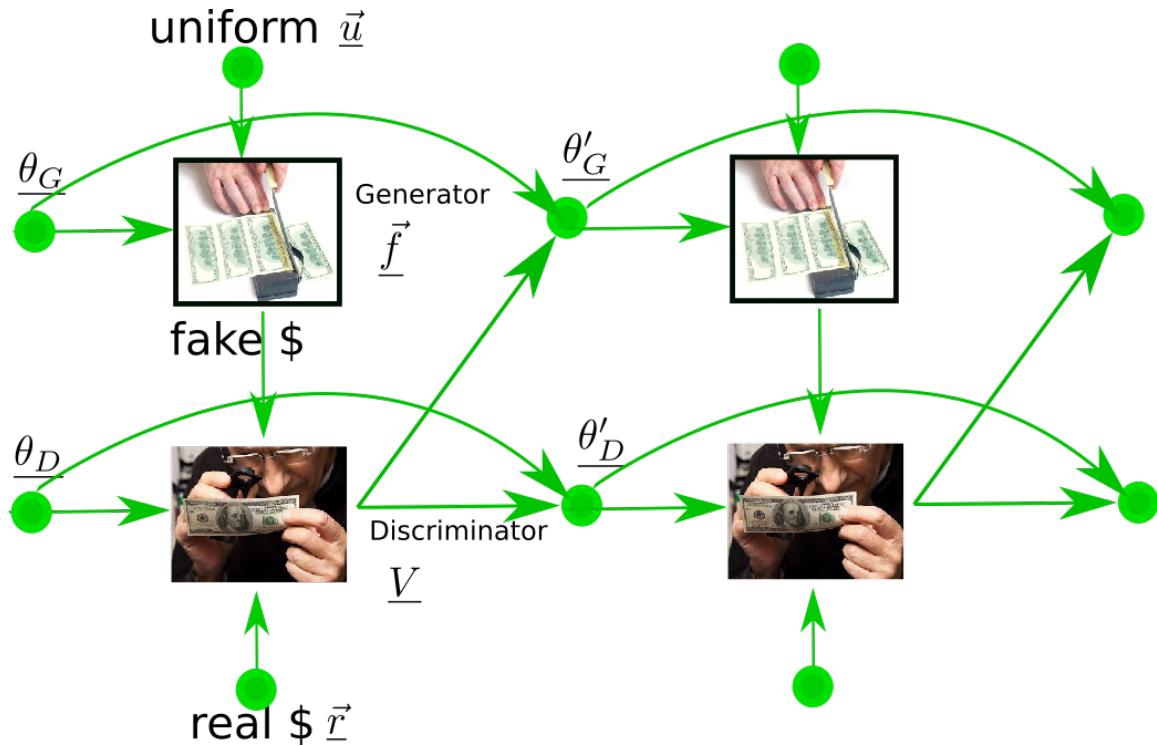


Figure 21.1: Generative Adversarial Network (GAN)

Original GAN, Ref.[9](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 verdict $V \in [0, 1]$. G depends on parameter θ_G and D on parameter θ_D . Verdict V and initial θ_G, θ_D are used to get new parameters θ'_G, θ'_D . Process is

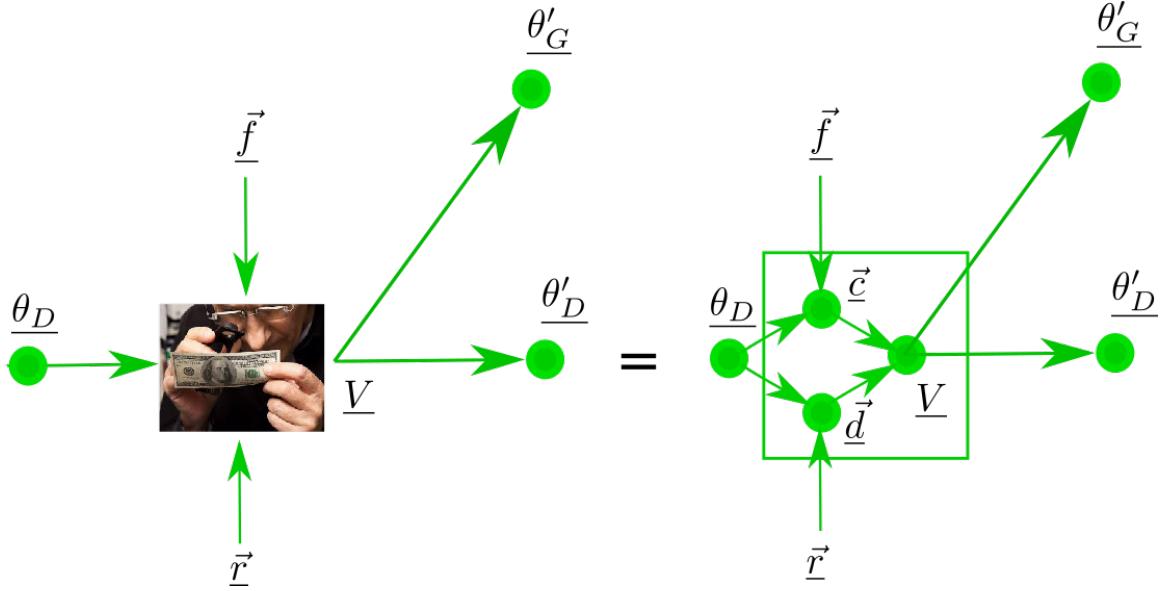


Figure 21.2: Discriminator node \underline{V} in Fig.21.1 can be split into 3 nodes \vec{c} , \vec{d} and \underline{V} .

repeated (Dynamical Bayesian Network) until saddle point in $V(\theta_G, \theta_D)$ 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 $x \in \mathcal{D}$,

$$0 \leq D(x, \theta_D) \leq 1 . \quad (21.1)$$

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

$$\sum_{x \in S} D(x, \theta_D) = \lambda(S, \theta_D) . \quad (21.2)$$

In general, $G(\cdot, \theta_G)$ need not be real valued.

Assume that for every $u \in S_u$, $G(u, \theta_G) = f \in S_f \subset \mathcal{D}$. Define

$$\overline{D}(f, \theta_D) = 1 - D(f, \theta_D) . \quad (21.3)$$

Note that

$$0 \leq \overline{D}(f, \theta_D) \leq 1 . \quad (21.4)$$

Define:

$$V(\theta_G, \theta_D) = \sum_r P(r) \ln D(r, \theta_D) + \sum_u P(u) \ln \overline{D}(G(u, \theta_G), \theta_D) . \quad (21.5)$$

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

$$\delta V(\theta_G, \theta_D) = 0 . \quad (21.6)$$

This implies

$$\partial_{\theta_G} V(\theta_G, \theta_D) = \partial_{\theta_D} V(\theta_G, \theta_D) = 0 \quad (21.7)$$

and

$$V_{opt} = \min_{\theta_G} \max_{\theta_D} V(\theta_G, \theta_D) . \quad (21.8)$$

Node TPMs for Figs.21.1 and 21.2 are given next in blue:

$$P(\theta_G) = \text{given} \quad (21.9)$$

$$P(\theta_D) = \text{given} \quad (21.10)$$

$$P(\vec{u}) = \prod_i P(u[i]) \quad (\text{usually uniform distribution}) \quad (21.11)$$

$$P(\vec{r}) = \prod_i P(r[i]) \quad (21.12)$$

$$P(f[i] | \vec{u}, \theta_G) = \delta[f[i], G(u[i], \theta_G)] \quad (21.13)$$

$$P(c[i] | \vec{f}, \theta_D) = \delta(c[i], \overline{D}(f[i], \theta_D)) \quad (21.14)$$

$$P(d[j] | \vec{r}, \theta_D) = \delta(d[j], D(r[j], \theta_D)) \quad (21.15)$$

$$P(V | \vec{d}, \vec{c}) = \delta(V, \frac{1}{N} \ln \prod_{i,j} (c[i] d[j])) \quad (21.16)$$

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 .

$$P(\theta'_G | V, \theta_G) = \delta(\theta'_G, \theta_G - \eta_G \partial_{\theta_G} V) \quad (21.17)$$

$$P(\theta'_D | V, \theta_D) = \delta(\theta'_D, \theta_D + \eta_D \partial_{\theta_D} V) \quad (21.18)$$

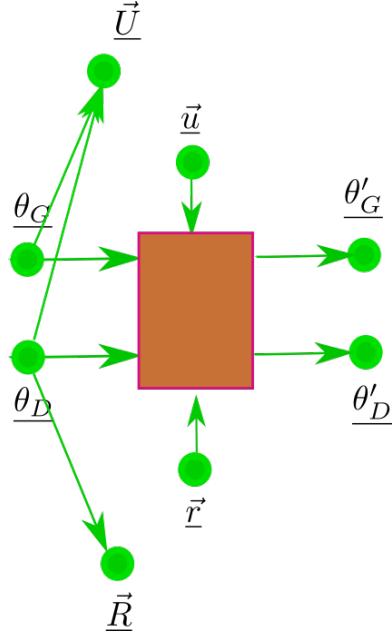


Figure 21.3: GAN, Constraining Bayesian Network

Constraining Bnet given in Fig.21.3. It adds 2 new nodes, namely \vec{U} and \vec{R} , to the bnet of Fig.21.1. 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.

$$P(U[i] | \theta_G) = \frac{\overline{D}(G(U[i], \theta_G), \theta_D))}{\lambda(\theta_G, \theta_D)} \quad (21.19)$$

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

$$P(R[i] | \theta_G, \theta_D) = \frac{D(R[i], \theta_D)}{\lambda(\theta_D)} \quad (21.20)$$

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

$$P(V|\vec{u}, \vec{r}) = \delta(V, \frac{1}{N} \ln \prod_{i,j} (P(R[i] = r[i] | \theta_G, \theta_D) P(U[i] = u[j] | \theta_G))) \quad (21.21)$$

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

\mathcal{L} = likelihood

$$\mathcal{L} = P(\vec{r}, \vec{u} | \theta_G, \theta_D) \quad (21.22)$$

$$= \prod_{i,j} \left[\frac{D(r[i], \theta_D)}{\lambda(\theta_D)} \frac{\overline{D}(G(u[j], \theta_G), \theta_D))}{\overline{\lambda}(\theta_G, \theta_D)} \right] \quad (21.23)$$

$$\ln \mathcal{L} = N[V(\theta_G, \theta_D) - \ln \lambda(\theta_D) - \ln \bar{\lambda}(\theta_G, \theta_D)] \quad (21.24)$$

Chapter 22

Goodness of Causal Fit

This chapter is an almost verbatim copy of a paper of mine, Ref.[41].

22.1 Abstract

We propose a Goodness of Causal Fit (GCF) measure which depends on Pearl “do” interventions. This is different from a measure of Goodness of Fit (GF), which does not use interventions. Given a DAG set \mathcal{G} , to find a good $G \in \mathcal{G}$, we propose plotting $GCF(G)$ versus $GF(G)$ for all $G \in \mathcal{G}$, and finding a graph $G \in \mathcal{G}$ with a large amount of both types of goodness.

22.2 Introduction

Frequently, when students first encounter Bayesian Networks (bnets) and Causal Inference (CI), they experience serious doubts about the usefulness of this theory, because they believe finding the underlying model (i.e., DAG) for most realistic physical situations is too difficult or impossible. I think that part of the problem is that these students are assuming, perhaps unconsciously, that there exists a unique DAG that fits Nature perfectly, and a mind-boggling number of possibilities to sift through to find that DAG. Rather than looking for a unique DAG, I think a better strategy is to write down a set \mathcal{G} of likely DAGs, and to calculate for each DAG in \mathcal{G} , a measure called Goodness of Causal Fit (GCF). Then use the DAGs with the highest GCF scores.

The goal of this paper is to propose a GCF measure. Such a measure is of course not unique, and someone may propose in the future a measure that is better than ours.

When designing a GCF measure, it is important to keep in mind the First Dictum¹ of CI: The data is model-less. In the First Dictum, when we say “data”, we are referring to what is commonly called a dataset. A dataset is a table of data,

¹ This is just my whimsical name for it.

where all the entries of each column have the same units, and measure a single feature, and each row refers to one particular sample or individual. Datasets are particularly useful for estimating probability distributions and for training neural nets. In the First Dictum, when we say “model”, we are referring to a DAG (directed acyclic graph) or a bnet (Bayesian Network= DAG + probability table for each node of DAG).

You can try to derive a model from a dataset, but you’ll soon find out that you can only go so far. The process of finding a *partial* model from a dataset is called structure learning (SL). SL can be done quite nicely with Marco Scutari’s open source program **bnlearn** (Ref[34]). The problem is that SL often cannot narrow down the model to a single one. It finds an undirected graph (UG), and it can determine the direction of some of the arrows in the UG, but it is often incapable, for well understood fundamental—not just technical—reasons, of finding the direction of *all* the arrows of the UG. So it often fails to fully specify a DAG model.

Let’s call the ordered pair (dataset, model) a **data SetMo**. Then what the First Dictum is saying is that a dataset is model-free or model-less (although sometimes one can find a partial model hidden in there). A dataset is not a data SetMo.

Graphs which contain both directed and undirected edges are called **partially directed (PD) graphs**. **bnlearn** takes a dataset as input and returns a PD graph G_{pd} . Given a PD graph G_{pd} , let $\mathcal{G}(G_{pd})$ be the DAG set \mathcal{G} which is generated by giving directions to all undirected edges of G_{pd} in all possible ways. We will refer to $\mathcal{G}(G_{pd})$ as the **DAG set generated by G_{pd}** , and to any $\mathcal{G}' \subset \mathcal{G}(G_{pd})$ as a **DAG set partially generated by G_{pd}** . Once we define below our GCF measure, we will apply it to sets of the type $\mathcal{G}(G_{pd})$ as an example.

It’s clear that any measure of GCF will have to involve interventions such as the “do” intervention invented by Pearl et al for CI. Without interventions like “do”, it is impossible to distinguish causally the DAGs in a set $\mathcal{G}(G_{pd})$.

Henceforth, random variables will be indicated by underlining.

22.3 Goodness of Fit

Before trying to define a GCF measure, it is instructive to review the closely related, well established, measures of Goodness of Fit (GF).

Consider two probability distributions $PO(x)$ and $PE(x)$, where $x \in S_{\underline{x}}$. By a GF measure, we mean a measure of the difference between PO and PE . Usually PO is the observed probability distribution and PE is the expected, theoretical one.

Three popular measures of the difference between PO and PE are:

1. The **Kullback-Liebler divergence**:

$$D_{KL}(PO \parallel PE) = \sum_{x \in S_{\underline{x}}} PO(x) \ln \frac{PO(x)}{PE(x)}. \quad (22.1)$$

2. The **Pearson divergence** (aka **Pearson Chi-squared test statistic**):

$$D_{\chi^2}(PO \parallel PE) = \sum_{x \in S_{\underline{x}}} \frac{[PO(x) - PE(x)]^2}{PE(x)} = \sum_{x \in S_{\underline{x}}} \frac{PO^2(x)}{PE(x)} - 1 . \quad (22.2)$$

It's easy to show using $\ln(1 + \delta) = \delta + \mathcal{O}(\delta^2)$ that if $\left| \frac{PO(x)}{PE(x)} - 1 \right| \ll 1$ for all $x \in S_{\underline{x}}$, then

$$D_{KL}(PO \parallel PE) \approx D_{\chi^2}(PO \parallel PE) \quad (22.3)$$

3. The **Euclidean distance squared**:

$$D_E(PO, PE) = \sum_{x \in S_{\underline{x}}} [PO(x) - PE(x)]^2 \quad (22.4)$$

Note that of these 3 measures, only $D_E(PO, PE)$ is symmetric in PO and PE .

Given any bnet G with full probability distribution² $P_G(x.)$ and a probability distribution³ $\tilde{P}(x.)$ derived empirically from a dataset, let

$$D(G) = \sum_{x.} \tilde{P}(x.) \ln \frac{\tilde{P}(x.)}{P_G(x.)} \quad (22.5)$$

$$= D_{KL}(\tilde{P}(\underline{x.}) \parallel P_G(\underline{x.})) \quad (22.6)$$

We define **Goodness of Fit (GF)** of the bnet G by

$$GF(G) = \ln \frac{1}{D(G)} \quad (22.7)$$

22.4 GCF example 1

For the first example of our measure of GCF, we consider $\mathcal{G}(G_{pd}) = \{G_1, G_2\}$ given by Fig.22.1. We will assume the following:

- First, we assume that we have collected a dataset from which we have extracted a full empirical distribution $\tilde{P}(z, a, b)$. From $\tilde{P}(z, a, b)$, we assume that the following have been calculated. $\tilde{P}(z, b|a)$ $\tilde{P}(z, a|b)$ $\tilde{P}(a)$, $\tilde{P}(b)$.
- Second, we assume that a dataset has been collected for which \underline{a} was held fixed to each of the possible values $a \in S_{\underline{a}}$ of \underline{a} . Furthermore, we assume that the distribution $\tilde{P}(z, b|do(\underline{a}) = a)$ has been extracted from that dataset.

²We define $x.$ to be a vector with components x_i

³ Empirical distributions will be denoted by P with a tilde over it.

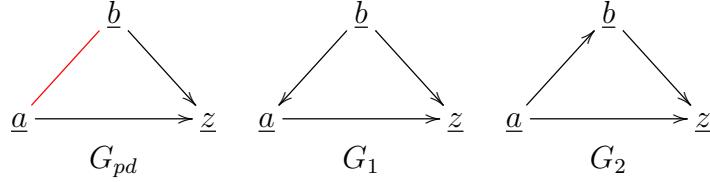


Figure 22.1: $\mathcal{G}(G_{pd}) = \{G_1, G_2\}$. The partially directed graph G_{pd} generates the DAGs G_1 and G_2 by giving directions to all undirected edges of G_{pd} in all possible ways. (In this case, there is only one undirected edge in G_{pd} .)

- Third, we assume that a dataset has been collected for which \underline{b} was held fixed to each of the possible values $b \in S_b$ of \underline{b} . Furthermore, we assume that the distribution $\tilde{P}(z, a | do(\underline{b}) = b)$ has been extracted from that dataset.

We will refer to $\tilde{P}(z, b | do(\underline{a}) = a)$ and $\tilde{P}(z, a | do(\underline{b}) = b)$ as **empirical do-probability distributions**.

Now define

$$D_a = \sum_{z,b} \tilde{P}(z, b | a) \ln \frac{\tilde{P}(z, b | a)}{\tilde{P}(z, b | do(\underline{a}) = a)} \quad (22.8)$$

$$= D_{KL}(\tilde{P}(z, \underline{b} | a) \| \tilde{P}(z, \underline{b} | do(\underline{a}) = a)) \quad (22.9)$$

$$D_{\underline{a}} = \sum_a \tilde{P}(a) D_a = E_a[D_a] \quad (22.10)$$

and

$$D_b = D_{KL}(\tilde{P}(z, \underline{a} | b) \| \tilde{P}(z, \underline{a} | do(\underline{b}) = b)) \quad (22.11)$$

$$D_{\underline{b}} = \sum_a \tilde{P}(b) D_b = E_b[D_b]. \quad (22.12)$$

We will refer to $D_{\underline{a}}$ and $D_{\underline{b}}$ as **do-divergences**.

Note that

$$D_a(G_2) = 0 \text{ for all } a \text{ so } \underbrace{D_{\underline{a}}(G_2)}_0 \leq D_{\underline{b}}(G_2) \quad (22.13)$$

and

$$D_b(G_1) = 0 \text{ for all } b \text{ so } D_{\underline{a}}(G_1) \geq \underbrace{D_{\underline{b}}(G_1)}_0. \quad (22.14)$$

If $D_{\underline{a}} \leq D_{\underline{b}}$, then $\underline{a} \rightarrow \underline{b}$, and if $D_{\underline{a}} \geq D_{\underline{b}}$ then $\underline{a} \leftarrow \underline{b}$. Thus, the arrow and the inequality sign point in opposite directions. Alternatively, just remember that the arrow points to the larger of the two D 's.

If $D_{\underline{a}} \leq D_{\underline{b}}$, then define $GCF(G_1) = -1$ and $GCF(G_2) = 1$.

If $D_{\underline{b}} \leq D_{\underline{a}}$, then define $GCF(G_1) = 1$, $GCF(G_2) = -1$.

22.5 GCF example 2

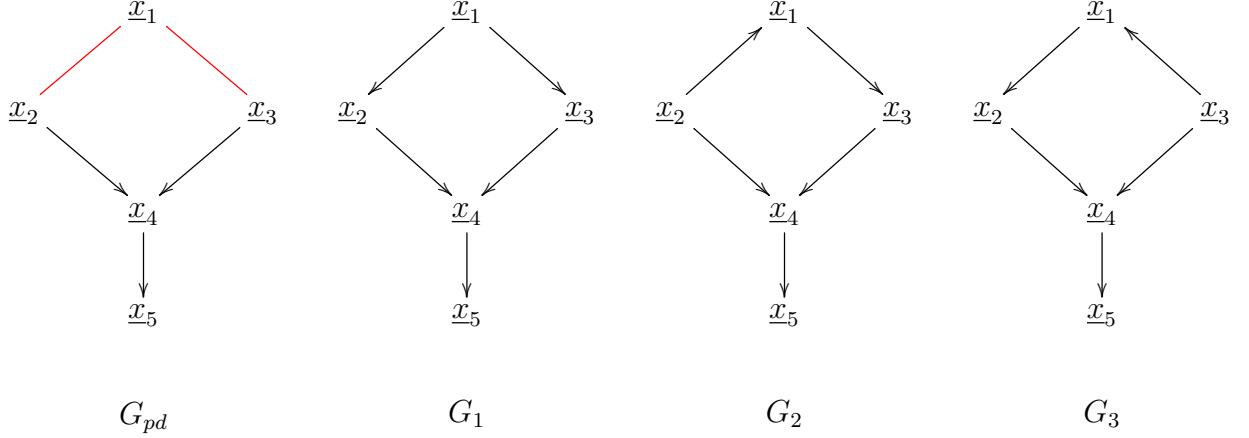


Figure 22.2: $\mathcal{G} = \{G_1, G_2, G_3\}$. \mathcal{G} is a set of observationally equivalent (OE) graphs. These are graphs that have the same value for GF, and are therefore indistinguishable from GF alone. For more info about OE graphs, see Chapter 42. Note that $\mathcal{G}(G_{pd})$ includes one more DAG, the one in which node \underline{x}_1 is a collider.

For the second example of our measure of GCF, consider $\mathcal{G} = \{G_1, G_2, G_3\}$ given by Fig.22.2.

Which of the do-divergences $D_{\underline{x}_2}$, $D_{\underline{x}_1}$ and $D_{\underline{x}_3}$, is smallest, middle and highest, depends on the empirical do-probability distributions. For definiteness, suppose the sizes of these do-divergences are related as follows:

$$D_{\underline{x}_2} \leq D_{\underline{x}_1} \leq D_{\underline{x}_3}. \quad (22.15)$$

For any two do-divergences $D_{\underline{a}}$ and $D_{\underline{b}}$, define the distance

$$d_{\underline{b}, \underline{a}} = |D_{\underline{b}} - D_{\underline{a}}| \quad (22.16)$$

If we abbreviate $D_{\underline{x}_j}$ by D_j , we can define the GCF for each of the graphs in \mathcal{G} by:

$$GCF(G_1) = \frac{-d_{2,1} + d_{1,3}}{d_{2,1} + d_{1,3}} \quad (22.17a)$$

$$GCF(G_2) = \frac{d_{2,1} + d_{1,3}}{d_{2,1} + d_{1,3}} = 1 \quad (22.17b)$$

$$GCF(G_3) = \frac{-d_{2,1} - d_{1,3}}{d_{2,1} + d_{1,3}} = -1 \quad (22.17c)$$

22.6 GCF in general

Eqs.(22.17) are a special case of the following formulas.

For any two do-divergences $D_{\underline{a}}$ and $D_{\underline{b}}$, define the **do-divergence distance** by

$$d_{\underline{b}, \underline{a}} = |D_{\underline{b}} - D_{\underline{a}}| . \quad (22.18)$$

For any $G_i \in \mathcal{G}$, define the **edge-sign function** by

$$\sigma_{G_i}(\underline{a} - \underline{b}) = \begin{cases} +1 & \text{if edge } \underline{a} - \underline{b} \text{ in } G_i \text{ points towards larger of } D_{\underline{a}} \text{ and } D_{\underline{b}}. \\ -1 & \text{otherwise} \end{cases} \quad (22.19)$$

Finally, suppose that \mathcal{G} is either partially or fully generated by a PD graph G_{pd} with undirected edges $\{\underline{a}_k - \underline{b}_k\}_{k=0,1,\dots,nk-1}$. Then define the GCF of graph $G_i \in \mathcal{G}$ by

$$GCF(G_i) = \frac{\sum_{k=0}^{nk-1} \sigma_{G_i}(\underline{a}_k - \underline{b}_k) d_{\underline{a}_k, \underline{b}_k}}{\sum_{k=0}^{nk-1} d_{\underline{a}_k, \underline{b}_k}} . \quad (22.20)$$

Note that $-1 \leq GCF(G_i) \leq 1$.

If the DAG set \mathcal{G} contains only one DAG G , define $GCF(G) = 1$, because all arrows in G are in the correct direction, as far as we know.

Call the **skeleton** of a DAG, the undirected graph that one obtains by turning all its edges from directed to undirected ones.

So far, we have applied our measure of GCF to a DAG set \mathcal{G} which is either fully or partially generated by a PD graph G_{pd} , or is a singleton set. But what if we want a GCF that can score every DAG in a DAG set \mathcal{G} that contains DAGs with different skeletons? In that case, let $\{\underline{a}_k - \underline{b}_k\}_{k=0,1,\dots,nk-1}$ be *all* the edges of graph $G_i \in \mathcal{G}$, and use the relative GCF given by Eq.(22.20), or use an absolute GCF defined by

$$GCF_a(G_i) = \sum_{k=0}^{nk-1} \sigma_{G_i}(\underline{a}_k - \underline{b}_k) d_{\underline{a}_k, \underline{b}_k} . \quad (22.21)$$

So let \mathcal{G} be an arbitrary DAG set. Our GCF (or GCF_a) measure is not enough to decide the best possible G in \mathcal{G} , because there might be several graphs with $GCF \approx 1$. For this reason, we recommend plotting $GCF(G)$ (or $GCF_a(G)$) versus $GF(G)$ for all $G \in \mathcal{G}$. Then choose a G with a large amount of both types of goodness. It might even be advantageous to average over a small subset of DAGs in \mathcal{G} that have large amounts of both types of goodness. This would be similar to averaging over an ensemble of decision trees to get a random forest.

A plot of $GCF(G)$ versus $GF(G)$ agrees with the spirit of the First Dictum and data setmos, because also in those, we acknowledge a separation between the dataset (GF) and model (GCF) degrees of freedom.

Chapter 23

Hidden Markov Model

A Hidden Markov Model (HMM) is a generalization of a Kalman Filter (KF). KFs are discussed in Chapter 29. The bnets of HMMs and KFs bnets are the same. The only difference is that a KF assumes special node TPMs.

See Wikipedia article Ref.[61] to learn about the history and many uses of HMMs. This chapter is based on Ref.[21].

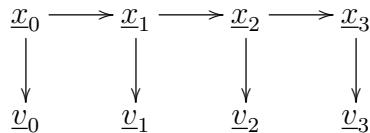


Figure 23.1: HMM bnet with $n = 4$.

Suppose

$v^n = (v_0, v_1, \dots, v_{n-1})$ are n visible nodes that are measured, and
 $x^n = (x_0, x_1, \dots, x_{n-1})$ are the n hidden, unmeasurable state nodes of a system
that is being monitored.

For the bnet of Fig.23.1, one has

$$P(x^n, v^n) = \prod_{i=0}^{n-1} P(x_i | x_{i-1}) P(v_i | x_i), \quad (23.1)$$

where $x_{-1} = 0$.

Let $x_{<i} = (x_0, x_1, \dots, x_{i-1})$.

For $i = 0, 1, \dots, n - 1$, define

\mathcal{F}_i =future measurements probability

$$\mathcal{F}_i(x_i) = P(v_{>i} | x_i) \quad (23.2)$$

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

$$\bar{\mathcal{F}}_i(x_i) = P(v_{<i}, v_i, x_i) \quad (23.3)$$

λ_i = present measurement probability

$$\lambda_i(x_i) = P(v_i|x_i) \quad (23.4)$$

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

$$\mathcal{F}_i(x_i) = \frac{1}{P(x_i)} \sum_{x_{>i}} \quad x_i \longrightarrow x_{>i} \downarrow \\ v_{>i} \quad (23.5)$$

$$\bar{\mathcal{F}}_i(x_i) = \sum_{x_{<i}} \quad x_{<i} \longrightarrow x_i \downarrow \\ v_{<i} \quad v_i \quad (23.6)$$

$$\lambda_i(x_i) = \frac{1}{P(x_i)} \quad x_i \downarrow \\ v_i \quad (23.7)$$

Claim 16 For $i \geq 0$,

$$P(x_i, v^n) = \bar{\mathcal{F}}_i(x_i) \mathcal{F}_i(x_i) . \quad (23.8)$$

For $i > 0$,

$$P(x_{i-1}, x_i, v^n) = \bar{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i|x_{i-1}) \mathcal{F}_i(x_i) . \quad (23.9)$$

proof:

$$P(x_i, v^n) = \sum_{x_{<i}} \sum_{x_{>i}} P(x^n, v^n) \quad (23.10)$$

$$= \sum_{x_{<i}} \sum_{x_{>i}} P(x^n, v^n|x_i) P(x_i) \quad (23.11)$$

$$= \sum_{x_{<i}} \sum_{x_{>i}} P(x_{<i}, v_{<i}, v_i|x_i) P(x_{>i}, v_{>i}|x_i) P(x_i) \quad (23.12)$$

$$= P(v_{<i}, v_i|x_i) P(v_{>i}|x_i) P(x_i) \quad (23.13)$$

$$= \bar{\mathcal{F}}_i(x_i) \mathcal{F}_i(x_i) \quad (23.14)$$

$$P(x_{i-1}, x_i, v^n) = \sum_{x_{<i-1}} \sum_{x_{>i}} P(x^n, v^n) \quad (23.15)$$

$$= \sum_{x_{<i-1}} \sum_{x_{>i}} P(x^n, v^n | x_{i-1}, x_i) P(x_{i-1}, x_i) \quad (23.16)$$

$$= \sum_{x_{<i-1}} \sum_{x_{>i}} P(x_{<i-1}, v_{<i-1}, v_{i-1} | x_{i-1}) P(v_i | x_i) P(x_{i-1}, x_i) P(x_{>i}, v_{>i} | x_i) \quad (23.17)$$

$$= P(v_{<i-1}, v_{i-1} | x_{i-1}) P(v_i | x_i) P(x_{i-1}, x_i) P(v_{>i} | x_i) \quad (23.18)$$

$$= \bar{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) \quad (23.19)$$

QED

Claim 17 For $i > 0$, \mathcal{F}_i and $\bar{\mathcal{F}}_i$ can be calculated recursively as follows:

$$\bar{\mathcal{F}}_i(x_i) = \sum_{x_{i-1}} \bar{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \quad (23.20)$$

$$\mathcal{F}_{i-1}(x_{i-1}) = \sum_{x_i} \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) \quad (23.21)$$

proof:

$$\bar{\mathcal{F}}_i(x_i) \mathcal{F}_i(x_i) = P(x_i, v^n) \quad (23.22)$$

$$= \sum_{x_{i-1}} P(x_{i-1}, x_i, v^n) \quad (23.23)$$

$$= \sum_{x_{i-1}} \bar{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) \quad (23.24)$$

$$\bar{\mathcal{F}}_{i-1}(x_{i-1}) \mathcal{F}_{i-1}(x_{i-1}) = P(x_{i-1}, v^n) \quad (23.25)$$

$$= \sum_{x_i} P(x_{i-1}, x_i, v^n) \quad (23.26)$$

$$= \sum_{x_i} \bar{\mathcal{F}}_{i-1}(x_{i-1}) \lambda_i(x_i) P(x_i | x_{i-1}) \mathcal{F}_i(x_i) \quad (23.27)$$

QED

Claim 18

$$P(x_i|x_{i-1}, v^n) = \frac{\lambda_i(x_i)\mathcal{F}_i(x_i)}{\mathcal{F}_{i-1}(x_{i-1})} P(x_i|x_{i-1}) \quad (23.28)$$

$$P(x_{i-1}|x_i, v^n) = \frac{\lambda_i(x_i)\overline{\mathcal{F}}_{i-1}(x_{i-1})}{\overline{\mathcal{F}}_i(x_i)} P(x_i|x_{i-1}) \quad (23.29)$$

proof:

$$P(x_i|x_{i-1}, v^n) = \frac{P(x_{i-1}, x_i, v^n)}{P(x_{i-1}, v^n)} \quad (23.30)$$

$$= \frac{\overline{\mathcal{F}}_{i-1}(x_{i-1})\lambda_i(x_i)P(x_i|x_{i-1})\mathcal{F}_i(x_i)}{\overline{\mathcal{F}}_{i-1}(x_{i-1})\mathcal{F}_{i-1}(x_{i-1})} \quad (23.31)$$

Analogous proof for Eq.(23.29).

QED

Chapter 24

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 $U : S_x \rightarrow \mathbb{R}$ called the **utility function**. Then the **expected utility** is defined as

$$E_U[U] = \sum_U P(U)U \quad (24.1)$$

$$= \sum_x \sum_U \underbrace{P(U|x)}_{\delta[U,U(x)]} P(x)U \quad (24.2)$$

$$= \sum_x P(x)U(x) . \quad (24.3)$$

An **utility node** can be understood as a node composed of 3 simpler bnet nodes. This is illustrated in Fig.24.1.

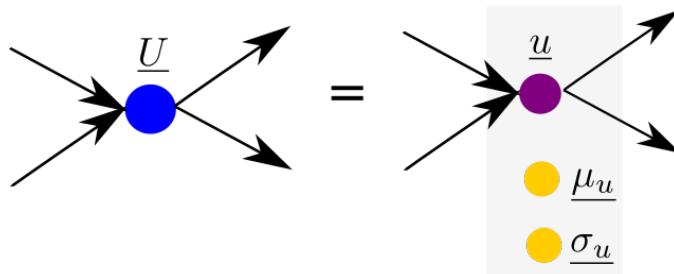


Figure 24.1: An utility node can be understood as a node composed of 3 simpler bnet nodes.

The TPMs, printed in blue, for the nodes of Fig.24.1, are as follows:

$$P(U|pa(U)) = \delta[U, U(pa(U))] , \quad (24.4)$$

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

$$P(u|pa(U)) = \delta[u, U(pa(U))] \quad (24.5)$$

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

$$P(\mu_u) = \delta(\mu_u, E_{\underline{u}}[\underline{u}]) \quad (24.6)$$

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

$$P(\sigma_u) = \delta(\sigma_u, \sqrt{E_{\underline{u}}[(\underline{u} - E_{\underline{u}}[\underline{u}])^2]}) \quad (24.7)$$

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.24.2. An influence diagram may have multiple utility nodes (\underline{U}_1 and \underline{U}_2 in Fig.24.2). Then one can define a merging utility node \underline{U} that sums the values of all the other utility nodes.

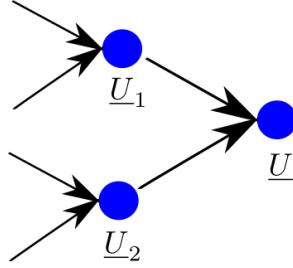


Figure 24.2: An influence diagram may have multiple utility nodes, say \underline{U}_1 and \underline{U}_2 . Then one can define an utility node $\underline{U} = \underline{U}_1 + \underline{U}_2$.

For the node \underline{U} of Fig.24.2,

$$P(\underline{U}|U_1, U_2) = \delta(\underline{U}, \underline{U}_1 + \underline{U}_2) \quad (24.8)$$

Chapter 25

Instrumental Inequality and beyond

This chapter is based on Refs. [4] and [24].

Instrumental Variables (IVs) are discussed in Chapter 26. This chapter will discuss the original Instrumental inequality (I-inequality) discovered by Pearl, and other related inequalities. The I-inequality arises in bnets that use an IV. The I-inequality bounds the effect that an IV \underline{z} can have on the outcome \underline{y} of a treatment $\underline{d} \rightarrow \underline{y}$. Since there is a path $\underline{z} \rightarrow \underline{d} \rightarrow \underline{y}$, the treatment dose \underline{d} acts as a mediator between the IV \underline{z} and the treatment outcome \underline{y} . The I-inequality is reminiscent of the data processing inequality $H(\underline{z} : \underline{y}) \leq H(\underline{d} : \underline{y})$ which is valid for a simple Markov chain bnet $\underline{z} \rightarrow \underline{d} \rightarrow \underline{y}$. The data processing inequality is saying that the endpoint \underline{y} receives more information from \underline{d} than from \underline{z} . This is reasonable, since \underline{y} is “closer” to \underline{d} than to \underline{z} .

25.1 I-inequality

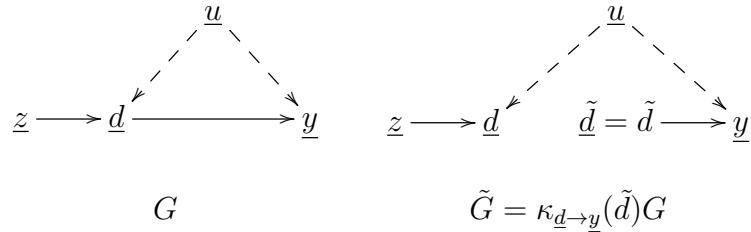


Figure 25.1: In bnet G , an IV \underline{z} acts on a treatment $\underline{d} \rightarrow \underline{y}$. Bnet \tilde{G} is obtained by applying an imagine operator to arrow $\underline{d} \rightarrow \underline{y}$ of bnet G .

Claim 19 *The TPMs for the bnet G in Fig.25.1 satisfy*

$$\max_d \sum_y \max_z P(d, y|z) \leq 1 \quad (25.1)$$

proof:

Below, any probability that alludes to a value \tilde{d} refers to bnet \tilde{G} . Otherwise, if it doesn't allude to \tilde{d} , then it refers to G (or to \tilde{G} , since the TPMs of \tilde{G} are defined from those of G in a consistent manner.)

G satisfies

$$P(d, y|z) = \sum_u P(u)P(y|u, d)P(d|u, z), \quad (25.2)$$

and \tilde{G} satisfies

$$P(d, y|z, \tilde{d}) = \sum_u P(u)P(y|u, \tilde{d})P(d|u, z). \quad (25.3)$$

Note that Eqs.(25.2) and (25.3) imply that

$$P(d, y|z, d) = P(d, y|z) \quad (25.4)$$

and that

$P(\tilde{d}, y|z, \tilde{d}) \leq \sum_d P(d, y|z, \tilde{d}) = P(y|\tilde{d}).$

(25.5)

Thus,

$$\max_{\tilde{d}} \sum_y \max_z P(\tilde{d}, y|z, \tilde{d}) \leq \max_{\tilde{d}} \sum_y \max_z P(y|\tilde{d}) \quad (25.6)$$

$$\leq \max_{\tilde{d}} \sum_y P(y|\tilde{d}) \quad (25.7)$$

$$\leq \max_{\tilde{d}} 1 \quad (25.8)$$

$$\leq 1 \quad (25.9)$$

QED

As pointed out in Ref.[4] from which I learned the above proof, the above proof is highly generalizable.

Fig.25.2 gives a graphical representation of the boxed Eq.(25.5) which is crucial to the proof.

And here is a meta-description of the steps in the proof:

$$\sum_u \begin{array}{c} \xrightarrow{\quad \underline{d} = \tilde{d} \quad} \\ \xrightarrow{\quad \underline{d} = \tilde{d} \quad} \end{array} \xrightarrow{\quad \underline{y} \quad} \leq \sum_d \sum_u \begin{array}{c} \xrightarrow{\quad \underline{d} = \tilde{d} \quad} \\ \xrightarrow{\quad \underline{d} = \tilde{d} \quad} \end{array} \xrightarrow{\quad \underline{y} \quad} = \quad .$$

Figure 25.2: Graphical representation of the boxed equation Eq.(25.5).

1. Use imagine operator to create a non-negative matrix $M_{d,\tilde{d}}$.
2. Use fact that row or column sum of $M_{d,\tilde{d}}$ is larger than diagonal element in sum:
 $\sum_d M_{d,\tilde{d}} \geq M_{\tilde{d},\tilde{d}}$.

25.1.1 I-inequality for binary $\underline{z}, \underline{d}, \underline{y}$

It is enlightening to write down the I-inequality for the special case that $\underline{z}, \underline{d}, \underline{y}$ are binary.

$$P(d=1, y|z) = \begin{matrix} & z=0 & z=1 \\ \begin{matrix} & y=0 \\ P(d=1, y|z) = & \end{matrix} & \begin{array}{|c|c|} \hline A & B \\ \hline C & D \\ \hline \end{array} \\ & y=1 & \end{matrix}$$

$$\begin{aligned} A + D &\leq 1 \\ B + C &\leq 1 \end{aligned}$$

Figure 25.3: I-inequality for binary $\underline{z}, \underline{d}, \underline{y}$. The same picture except with $d = 0$ is also true.

In the binary case, the I-inequality implies 4 different inequalities. These are as follows. One gets two inequalities by setting $d = 1$ in the next 2 equations.

$$\sum_{y=0}^1 \sum_{z=0}^1 \mathbb{1}(y=z) P(d, y|z) , \quad (25.10a)$$

$$\sum_{y=0}^1 \sum_{z=0}^1 \mathbb{1}(y \neq z) P(d, y|z) . \quad (25.10b)$$

One gets an additional 2 inequalities by setting $d = 0$ in Eqs.(25.10). These 4 inequalities are illustrated in Fig.25.3.

What do they mean? That at fixed \underline{d} , the correlation between \underline{z} and \underline{y} is limited.

25.2 Bounds on Effect of IV on treatment outcome \mathbf{y}

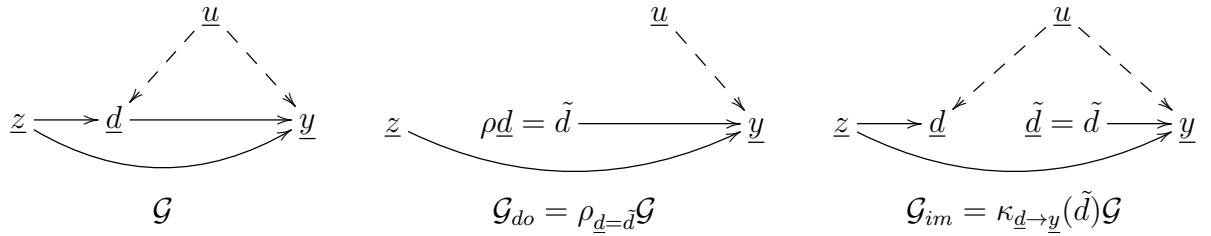


Figure 25.4: Bnet \mathcal{G} is obtained from the bnet G in Fig.25.1 by adding to G an arrow from the IV \underline{z} to the treatment outcome \underline{y} . Bnet \mathcal{G}_{do} is obtained by applying a do operator to node \underline{d} of \mathcal{G} . Bnet \mathcal{G}_{im} is obtained by applying an imagine operator to arrow $\underline{d} \rightarrow \underline{y}$ of \mathcal{G} .

In this section, we will assume that random variables $\underline{z}, \underline{d}, \underline{y}$ are binary. Just as with the binary case of the I-inequality, we will find an inequality for each value of $\underline{d} \in \{0, 1\}$.

Below, we will use the following 3 shorthand notations:

$$P_{y|z}(d) = P(d, y|z) , \quad (25.11)$$

$$P_{|z}(d) = \sum_y P(d, y|z) , \quad (25.12)$$

and

$$\pi_{|z}(d) = 1 - P_{|z}(d) . \quad (25.13)$$

For the bnet \mathcal{G}_{do} in Fig.25.4, define the IV effect at fixed $\rho\underline{d} = \tilde{d}$ by

$$IVE(\tilde{d}) = P(y = 1|z = 1, \rho\underline{d} = \tilde{d}) - P(y = 1|z = 0, \rho\underline{d} = \tilde{d}) . \quad (25.14)$$

Claim 20 *The TPMs for the bnet \mathcal{G}_{do} in Fig.25.4 satisfy*

$$\pi_{|0}(d) \leq [IVE(d) - \{P_{1|1}(d) - P_{1|0}(d)\}] \leq \pi_{|1}(d) \quad (25.15)$$

proof:

$$P(y|z, \rho \underline{d} = \tilde{d}) = \sum_u P(u) P(y|u, z, \tilde{d}) \quad (25.16)$$

$$= \sum_u P(u) \sum_d P(d, y|u, z, \tilde{d}) \quad (25.17)$$

$$\geq \sum_u P(u) P(\tilde{d}, y|u, z, \tilde{d}) \quad (25.18)$$

$$= \sum_u P(u) P(\tilde{d}, y|u, z) \quad (25.19)$$

$$= P_{y|z}(\tilde{d}) \quad (25.20)$$

Next note that $P(d, y|z, \tilde{d}) \geq 0$, and $\sum_{d,y} P(d, y|z, \tilde{d}) = 1$. If we write a table for $P(d, y|z, \tilde{d})$ at fixed z, \tilde{d} with row and column indices (d, y) , then a partial sum of the entries of that table must be ≤ 1 :

$$\sum_{d \neq \tilde{d}} P(d, y|z, \tilde{d}) + \underbrace{\sum_{y'} P(\tilde{d}, y'|z, \tilde{d})}_{P_{|z}(\tilde{d})} \leq 1 . \quad (25.21)$$

Using the definitions of $P_{|z}$ and $\pi_{|z}$, we can rewrite the last equation as

$$\sum_{d \neq \tilde{d}} P(d, y|z, \tilde{d}) \leq \pi_{|z}(\tilde{d}) . \quad (25.22)$$

Next note that

$$P(y|z, \rho \underline{d} = \tilde{d}) = \sum_u P(u) P(y|u, z, \tilde{d}) \quad (25.23)$$

$$= \sum_u P(u) \sum_d P(d, y|u, z, \tilde{d}) \quad (25.24)$$

$$= P(\tilde{d}, y|z, \tilde{d}) + \sum_{d \neq \tilde{d}} P(d, y|z, \tilde{d}) \quad (25.25)$$

$$= P_{y|z}(\tilde{d}) + \sum_{d \neq \tilde{d}} P(d, y|z, \tilde{d}) \quad (25.26)$$

$$\leq P_{y|z}(\tilde{d}) + \pi_{|z}(\tilde{d}) . \quad (25.27)$$

Hence,

$$P_{y|z}(\tilde{d}) \leq P(y|z, \rho \underline{d} = \tilde{d}) \leq P_{y|z}(\tilde{d}) + \pi_{|z} \quad (25.28)$$

$$P_{1|1}(\tilde{d}) \leq P(y=1|z=1, \rho \underline{d} = \tilde{d}) \leq P_{1|1}(\tilde{d}) + \pi_{|1} \quad (25.29)$$

$$-P_{1|0}(\tilde{d}) - \pi_{|0} \leq -P(y=1|z=0, \rho \underline{d} = \tilde{d}) \leq -P_{1|0}(\tilde{d}) \quad (25.30)$$

QED

Chapter 26

Instrumental Variables

This chapter is based on Refs.[3] and [63].

The theory of potential outcomes (PO) discussed in Chapter 43 assumes that confounders can be ignored by conditioning on them. However, there are cases when that is not possible, as when there are some unmeasured (i.e., unobserved, hidden) confounder nodes in the bnet, because one can only condition on observed random variables, by definition. So what if confounders can't be ignored? Are we then precluded from using PO theory? Not necessarily. It might still be possible to use PO theory if one can find a suitable instrumental variable (IV) for the problem.

IVs were actually invented by Sewall Wright and his father Philip Wright long before PO theory was invented by Rubin. The reason why IVs save PO theory is greatly clarified by using Pearl causal DAGs and his d-separation theorem (see Chapter 16).

Most of the discussion in this chapter is limited to LDEN (linear deterministic bnets with external noise). These are discussed in Chapter 31. However, as will become obvious to the reader, IVs are also applicable and useful in general bnet modeling.

26.1 δ with unmeasured confounder

In this section, we explain using LDENs why unmeasured confounders prejudice PO calculations.

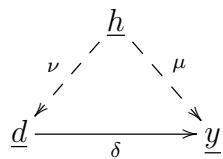


Figure 26.1: An LDEN bnet. The direct path $d \rightarrow y$ is confounded by a hidden variable h .

Consider the LDEN bnet of Fig.26.1, For some $\delta, \mu \in \mathbb{R}$, we have

$$\underline{y} = \delta \underline{d} + \underbrace{\mu \underline{h} + \underline{u}_y}_{\underline{n}_y}. \quad (26.1)$$

If $\langle \underline{n}_y, \underline{d} \rangle = 0$, then

$$\langle \underline{y}, \underline{d} \rangle = \delta_0 \langle \underline{d}, \underline{d} \rangle, \quad (26.2)$$

whereas if $\langle \underline{n}_y, \underline{d} \rangle \neq 0$, then

$$\langle \underline{y}, \underline{d} \rangle = \delta_1 \langle \underline{d}, \underline{d} \rangle + \langle \underline{n}_y, \underline{d} \rangle. \quad (26.3)$$

Therefore,

$$\delta_0 = \frac{\langle \underline{y}, \underline{d} \rangle}{\langle \underline{d}, \underline{d} \rangle}, \quad (26.4)$$

$$\delta_1 = \underbrace{\frac{\langle \underline{y}, \underline{d} \rangle}{\langle \underline{d}, \underline{d} \rangle}}_{\delta_0} - \frac{\langle \underline{n}_y, \underline{d} \rangle}{\langle \underline{d}, \underline{d} \rangle}. \quad (26.5)$$

If we assume no confounders and there is one, this gives the difference between the estimate δ_1 of δ for the truth, versus the naive estimate δ_0 .

If the confounder \underline{h} had been measured, then we would calculate the covariances at fixed \underline{n}_y , and the conditional covariance $\langle \underline{n}_y, \underline{d} \rangle|_{\underline{n}_y} = 0$

26.2 δ (with unmeasured confounder) can be inferred via IV

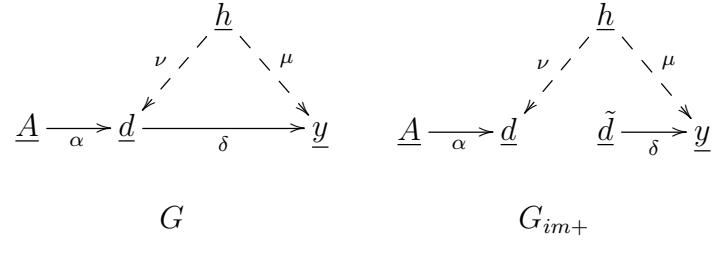


Figure 26.2: Two LDEN bnets. The direct path $\underline{d} \rightarrow \underline{y}$ is confounded by a hidden variable \underline{h} , but by using the IV \underline{A} , we are still able to identify (i.e. calculate) δ .

Now consider the two LDEN bnets shown in Fig.26.2. Note that there are no arrows $\underline{A} \rightarrow \underline{y}$ or $\underline{A} \rightarrow \underline{h}$. Note that node \underline{d} is a collider in the path $\underline{A} - \underline{d} - \underline{h} - \underline{y}$. Therefore, the only unblocked path from \underline{A} to \underline{y} in G is $\underline{A} \rightarrow \underline{d} \rightarrow \underline{y}$ and that path has been removed in G_{im+} . These observations are encapsulated in the following statements.

$$\underline{d} \perp_G \underline{y} = \text{false}, \quad \underline{A} \perp_G \underline{y} = \text{false}. \quad (26.6)$$

$$\underline{d} \perp_{G_{im+}} \underline{y} = \text{false}, \quad \underline{A} \perp_{G_{im+}} \underline{y} = \text{true}. \quad (26.7)$$

The following is true for G :

$$\underline{y} = \delta \underline{d} + \underbrace{\mu \underline{h} + \underline{u}_y}_{n_y} \quad (26.8)$$

$$\underline{d} = \alpha \underline{A} + \underbrace{\nu \underline{h} + \underline{u}_d}_{n_d}. \quad (26.9)$$

Since $\langle n_y, \underline{A} \rangle = \langle n_d, \underline{A} \rangle = 0$ is true in G , we have

$$\langle \underline{y}, \underline{A} \rangle = \delta \langle \underline{d}, \underline{A} \rangle \quad (26.10)$$

and

$$\langle \underline{d}, \underline{A} \rangle = \alpha \langle \underline{A}, \underline{A} \rangle. \quad (26.11)$$

Note that $\langle \underline{y}, \underline{A} \rangle = \delta = 0$ for G_{im+} but not for G , so we are speaking about G from here on. It follows that

$$\alpha = \frac{\langle \underline{d}, \underline{A} \rangle}{\langle \underline{A}, \underline{A} \rangle} \quad (26.12)$$

and

$$\delta = \frac{\langle \underline{y}, \underline{A} \rangle}{\langle \underline{d}, \underline{A} \rangle} \quad (26.13)$$

$$= \frac{\langle \underline{y}, \underline{A} \rangle}{\langle \underline{A}, \underline{A} \rangle} \frac{\langle \underline{A}, \underline{A} \rangle}{\langle \underline{d}, \underline{A} \rangle} \quad (26.14)$$

$$= \frac{\langle \underline{y}, \underline{A} \rangle}{\langle \underline{A}, \underline{A} \rangle} \frac{1}{\alpha} \quad (26.15)$$

$$= \frac{\langle \underline{y}, \alpha \underline{A} \rangle}{\langle \alpha \underline{A}, \alpha \underline{A} \rangle}. \quad (26.16)$$

26.3 More general bnets with IVs

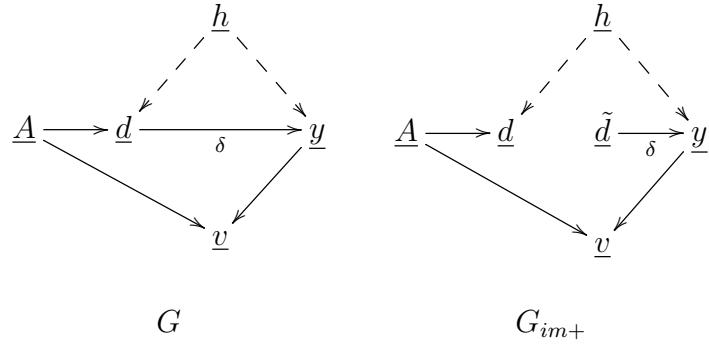


Figure 26.3: The 2 paths in G_{im+} from IV variable \underline{A} to \underline{y} are blocked by colliders \underline{v} and \underline{d} . Thus, $\underline{d} \perp_{G_{im+}} \underline{y} = \text{false}$, $\underline{A} \perp_{G_{im+}} \underline{y} = \text{true}$

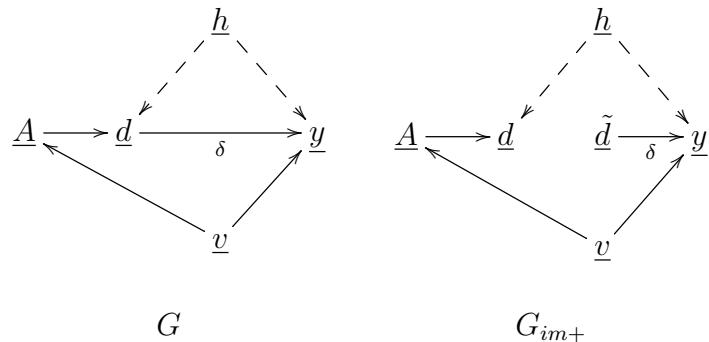


Figure 26.4: There are 2 paths in G_{im+} from IV variable \underline{A} to \underline{y} . One is blocked by the collider \underline{d} and the other can be blocked by conditioning on \underline{v} . Thus, $\underline{d} \perp_{G_{im+}} \underline{y} | \underline{v} = \text{false}$, $\underline{A} \perp_{G_{im+}} \underline{y} | \underline{v} = \text{true}$

Figs. 26.3 and 26.4 are examples of other bnets for which the effect δ is identifiable thanks to the IV \underline{A} .

26.4 Instrumental Inequality

Pearl's instrumental inequality and related inequalities are discussed in Chapter 25.

Chapter 27

Jackknife Resampling

This paper is based on Ref. [65].

Before reading this chapter, we recommend that you read the section entitled “Demystifying Population and Sample Variances”, in Chapter Notational Conventions and Preliminaries.

Jackknife Resampling (JR) is a way of generating from an original list of n samples, a new list of n synthetic (i.e., man made, not occurring physically in Nature) samples obtained by deleting one of the samples from the original list and averaging over the rest.

Let $\Sigma = \{0, 1, 2, \dots, n - 1\}$. Let us consider the list of samples

$$\vec{x} = (x^\sigma)_{\sigma \in \Sigma}, \quad (27.1)$$

where the x^σ are assumed to be i.i.d. with

$$E[\underline{x}^\sigma] = \mu \quad (27.2)$$

and

$$\left\langle \underline{x}^\sigma, \underline{x}^{\sigma'} \right\rangle = V_1 \delta(\sigma, \sigma'). \quad (27.3)$$

If we define μ and V_1 estimators by

$$\hat{\mu} = \frac{1}{n} \sum_{\sigma} x^\sigma \quad (27.4a)$$

$$\hat{V}_1 = \frac{1}{n-1} \sum_{\sigma} (x^\sigma - \hat{\mu})^2, \quad (27.4b)$$

then one can show that:

$$E[\hat{\mu}] = \mu, \quad E[\hat{V}_1] = V_1 \quad (27.5)$$

so both of these estimators are unbiased.

Now define lists of samples with one of the items in \vec{x} deleted:

$$\vec{x}_\xi = (x^\sigma)_{\sigma \in \Sigma - \{\xi\}} . \quad (27.6)$$

Suppose we are given functions of \vec{x} and \vec{x}_ξ

$$A = A^n(\vec{x}) , \quad (27.7)$$

$$A_\xi = A^{n-1}(\vec{x}_\xi) . \quad (27.8)$$

Then define a list \vec{A} by

$$\vec{A} = (A_\xi)_{\xi \in \Sigma} . \quad (27.9)$$

One can also define a list \vec{B} by:

$$\vec{B} = (B_\xi)_{\xi \in \Sigma} \quad (27.10)$$

where

$$B_\xi = nA - (n-1)A_\xi \quad (27.11)$$

$$= A_\xi - n[A_\xi - A] . \quad (27.12)$$

Later on, we will see why the list \vec{B} just defined is of interest.

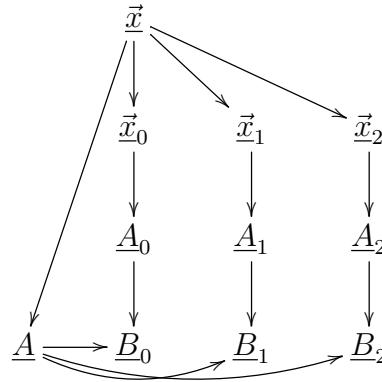


Figure 27.1: Bnet for jackknife resampling (JR).

Fig.27.1 is a bnet that encapsulates JR. The TPMs, printed in blue, for the nodes of that bnet, are as follows:

$$P(\vec{x}_\xi | \vec{x}) = \mathbb{1}(\vec{x}_\xi = \text{defined by Eq.(27.6)}) \quad (27.13)$$

$$P(A_\xi | \vec{x}_\xi) = \mathbb{1}(A_\xi = \text{defined by Eq.(27.8)}) \quad (27.14)$$

$$P(B_\xi | \vec{A}_\xi, A) = \mathbb{1}(\quad B_\xi = \text{ defined by Eq.(27.12)} \quad) \quad (27.15)$$

27.1 Case $A = A^n(\vec{x}) = \frac{1}{n} \sum_{\sigma} x^{\sigma}$

Suppose

$$A = \frac{1}{n} \underbrace{\sum_{\sigma} x^{\sigma}}_{E_{\sigma}[x^{\sigma}]} \quad (27.16)$$

and

$$A_\xi = \underbrace{\frac{1}{n-1} \sum_{\sigma \in \Sigma - \{\xi\}} x^{\sigma}}_{E_{\sigma|\xi}[x^{\sigma}] \text{ where } P(\sigma|\xi) = \frac{\mathbb{1}(\sigma \neq \xi)}{n-1}} . \quad (27.17)$$

Then

$$\frac{1}{n} \sum_{\xi} A_\xi = E_{\xi} E_{\sigma|\xi}[x^{\sigma}] = E_{\sigma}[x^{\sigma}] = A \quad (27.18)$$

Claim 21

$$E[\underline{A}_{\xi}] = \mu \quad (27.19)$$

$$\langle \underline{A}_{\xi}, \underline{A}_{\xi'} \rangle = V_1 \left[\frac{n-2-\delta(\xi, \xi')}{n-1} \right] \quad (27.20)$$

proof:

$$E[\underline{A}_{\xi}] = \frac{1}{n-1} \sum_{\sigma} \mathbb{1}(\xi \neq \sigma) E[x^{\sigma}] = \mu \quad (27.21)$$

$$\langle \underline{A}_{\xi}, \underline{A}_{\xi'} \rangle = \frac{1}{n-1} \sum_{\sigma} \sum_{\sigma'} [1 - \delta(\sigma, \xi)][1 - \delta(\sigma', \xi')] \langle x^{\sigma}, x^{\sigma'} \rangle \quad (27.22)$$

$$= \frac{V_1}{n-1} \sum_{\sigma} \sum_{\sigma'} [1 - \delta_{\xi}^{\sigma} - \delta_{\xi'}^{\sigma'} + \delta_{\xi}^{\sigma} \delta_{\xi'}^{\sigma'}] \delta_{\sigma'}^{\sigma} \quad (27.23)$$

$$= \frac{V_1}{n-1} [n-2 + \delta_{\xi'}^{\xi}] \quad (27.24)$$

QED

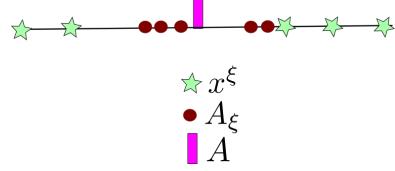


Figure 27.2: For each point x^ξ , there is a corresponding point A_ξ which is $n - 1$ times closer to the average A than x^ξ is.

Claim 22

$$A_\xi - A = \frac{A - x^\xi}{n - 1} \quad (27.25)$$

Hence, the distance of a point x^ξ to the mean value A is $n - 1$ times as large as the distance of A_ξ to A . (see Fig.27.2)

proof:

$$A_\xi - A = \frac{1}{n - 1} \left(\sum_{\sigma} x^\sigma - x^\xi \right) - \frac{1}{n} \sum_{\sigma} x^\sigma \quad (27.26)$$

$$= x^\xi \left(\frac{-1}{n - 1} \right) + \sum_{\sigma} x^\sigma \underbrace{\left(\frac{1}{n - 1} - \frac{1}{n} \right)}_{\frac{1}{n(n-1)}} \quad (27.27)$$

$$= \frac{A - x^\xi}{n - 1} \quad (27.28)$$

QED

Note that

$$(n - 1) \sum_{\xi} (A_\xi - E_{\xi}[A_\xi])^2 = \hat{V}_1 \quad (27.29)$$

by Eq.(27.25). Hence, we can estimate V_1 from \vec{A} instead of \vec{x} .

Note that

$$B_\xi = nA - (n - 1)A_\xi \quad (27.30)$$

$$= \sum_{\sigma} x^\sigma - \sum_{\sigma \neq \xi} x^\sigma = x^\xi \quad (27.31)$$

Since $B_\xi = x^\xi$, they have identical statistics. In particular, one can use for B_ξ the same μ and V_1 estimators that we defined in Eqs.(27.4) for x^σ .

Chapter 28

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 35) to the ensuing polytree. The first breakthrough paper to achieve this agenda in full was Ref.[14] by Lauritzen, and Spiegelhalter in 1988. See the Wikipedia article Ref.[66] 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.[12]. Ref.[12] is an excellent cookbook for programmers of the JT algo. My open source program QuantumFog (see Ref.[42]) implements the JT algo in Python, following the recipe of Ref.[12].

Chapter 29

Kalman Filter

A Kalman Filter is a special case of a Hidden Markov Model. HMMs are discussed in Chapter 23.

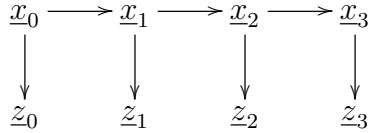


Figure 29.1: Kalman Filter bnet with $T = 4$.

Let $t = 0, 1, 2, \dots, T - 1$.

$\underline{x}_t \in S_x$ are random variables that represent the hidden (unobserved) true state of the system.

$\underline{z}_t \in S_z$ are random variables that represent the measured (observed) state of the system.

The Kalman Filter bnet Fig.29.1 has the following node TPMs, printed in blue:

$$P(x_t|x_{t-1}) = \mathcal{N}(x_t; F_t x_{t-1} + B_t u_t, Q_t), \quad (29.1)$$

where F_t, Q_t, B_t, u_t are given. $P(x_t|x_{t-1})$ becomes $P(x_t)$ for $t = 0$.

$$P(z_t|x_t) = \mathcal{N}(z_t; H_t x_t, R_t), \quad (29.2)$$

where H_t, R_t are given.

Define

$$\underline{Z}_t = (\underline{z}_{t'})_{t' \leq t}. \quad (29.3)$$

Define \hat{x}_t and P_t by

$$P(x_t|Z_t) = \mathcal{N}(x_t; \hat{x}_t, P_t). \quad (29.4)$$

29.1 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.29.2. For that figure,

$$P(\hat{x}_t, P_t | z_t, \hat{x}_{t-1}, P_{t-1}) = \delta(\hat{x}_t, ?) \delta(P_t, ?) . \quad (29.5)$$

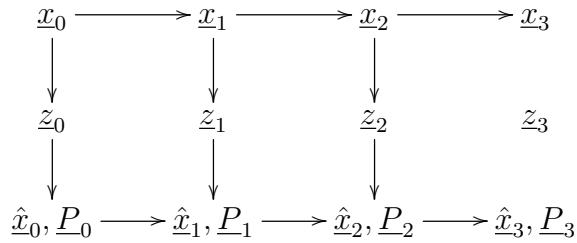


Figure 29.2: Kalman Filter bnet with deterministic nodes for \hat{x}_t, P_t .

29.2 Solution

Solution copied from Wikipedia Ref.[68].

Define $\eta_{t|t} = \eta_t$ for $\eta = \hat{x}, P$.

- **Predict**

Predicted (a priori) state estimate

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} + B_t u_t \quad (29.6)$$

Predicted (a priori) estimate covariance

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^\top + Q_t \quad (29.7)$$

- **Update**

Innovation (or measurement pre-fit residual)

$$\tilde{y}_{t|t-1} = z_t - H_t \hat{x}_{t|t-1} \quad (29.8)$$

Innovation (or pre-fit residual) covariance

$$S_t = H_t P_{t|t-1} H_t^\top + R_t \quad (29.9)$$

Optimal Kalman gain

$$K_t = P_{t|t-1} H_t^\top S_t^{-1} \quad (29.10)$$

Updated (a posteriori) state estimate

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t \tilde{y}_t \quad (29.11)$$

Updated (a posteriori) estimate covariance

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \quad (29.12)$$

Measurement post-fit residual

$$\tilde{y}_{t|t} = z_t - H_t \hat{x}_{t|t} \quad (29.13)$$

Chapter 30

Linear and Logistic Regression

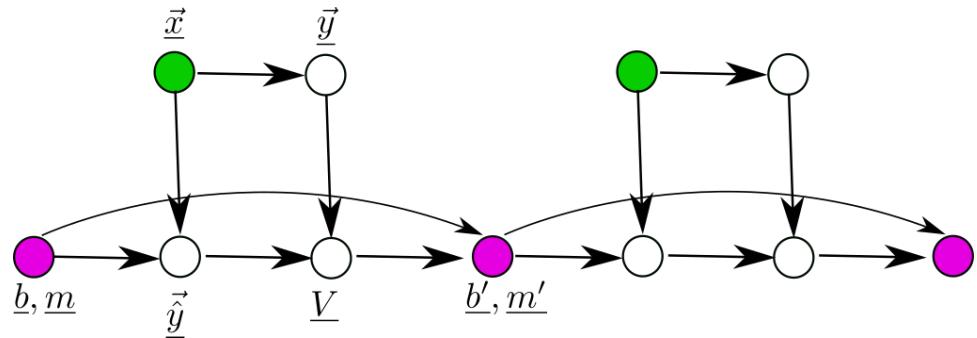


Figure 30.1: Linear Regression

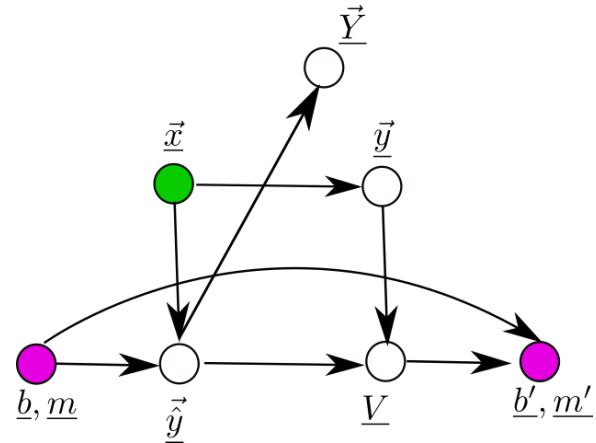


Figure 30.2: Bnet of Fig.30.1 with new \vec{Y} node.

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 .

$$\hat{y}(x; b, m) = b + mx \quad (30.1)$$

- **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 .

$$\hat{y}(x; b, m) = \text{smoid}(b + mx) \quad (30.2)$$

Define

$$V(b, m) = \sum_{x,y} P(x, y) |y - \hat{y}(x; b, m)|^2. \quad (30.3)$$

We want to minimize $V(b, m)$ (called a cost or loss function) wrt b and m .

Node TPMs of Bnet of Fig.30.1 given next in blue.

$$P(b, m) = \text{given} \quad (30.4)$$

The first time it is used, (b, m) is arbitrary. After the first time, it is determined by previous stage.

Let

$$P_{\underline{x}, \underline{y}}(x, y) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \mathbb{1}(x = x[\sigma], y = y[\sigma]). \quad (30.5)$$

$$P(\vec{x}) = \prod_{\sigma} P(x[\sigma]) \quad (30.6)$$

$$P(\vec{y}|\vec{x}) = \prod_{\sigma} P(y[\sigma] | x[\sigma]) \quad (30.7)$$

$$P(\vec{\hat{y}}|\vec{x}, b, m) = \prod_{\sigma} \delta(\hat{y}[\sigma], \hat{y}(x[\sigma], b, m)) \quad (30.8)$$

$$P(V|\vec{\hat{y}}, \vec{y}) = \delta(V, \frac{1}{nsam(\vec{x})} \sum_{\sigma} |y[\sigma] - \hat{y}[\sigma]|^2) \quad (30.9)$$

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”.

$$P(b'|V, b) = \delta(b', b - \eta_b \partial_b V) \quad (30.10)$$

$$P(m'|V, m) = \delta(m', m - \eta_m \partial_m V) \quad (30.11)$$

30.1 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:

$$x[\sigma] = (x_0[\sigma], x_1[\sigma], x_2[\sigma], \dots, x_{n-1}[\sigma]) . \quad (30.12)$$

Slope m is replaced by weights

$$w = (w_0, w_1, w_2, \dots, w_{n-1}) , \quad (30.13)$$

and the product of 2 scalars $mx[\sigma]$ is replaced by the inner vector product $w^T x[\sigma]$.

30.2 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

$$p[\sigma](1) = y[\sigma], \quad p[\sigma](0) = 1 - y[\sigma] \quad (30.14)$$

$$\hat{p}[\sigma](1) = \hat{y}[\sigma], \quad \hat{p}[\sigma](0) = 1 - \hat{y}[\sigma] \quad (30.15)$$

Then for logistic regression, the following 2 cost functions $V(b, m)$ can be used as alternatives to the cost function Eq.(30.3) previously given.

$$V(b, m) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} D_{KL}(p[\sigma] \parallel \hat{p}[\sigma]) \quad (30.16)$$

and

$$V(b, m) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} CE(p[\sigma] \rightarrow \hat{p}[\sigma]) \quad (30.17)$$

$$= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \{y[\sigma] \ln \hat{y}[\sigma] + (1 - y[\sigma]) \ln(1 - \hat{y}[\sigma])\} \quad (30.18)$$

$$= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \ln \{\hat{y}[\sigma]^{y[\sigma]} (1 - \hat{y}[\sigma])^{(1-y[\sigma])}\} \quad (30.19)$$

$$= \frac{-1}{nsam(\vec{x})} \sum_{\sigma} \ln P(Y = y[\sigma] \mid \hat{y} = \hat{y}[\sigma]) \quad (30.20)$$

$$= - \sum_{x,y} P(x, y) \ln P(Y = y \mid \hat{y} = \hat{y}(x, b, m)) \quad (30.21)$$

Above, we used

$$P(\underline{Y} = Y | \hat{y}) = \hat{y}^Y [1 - \hat{y}]^{1-Y} \quad (30.22)$$

for $Y \in S_{\underline{Y}} = \{0, 1\}$. (Bernoulli distribution).

There is no node corresponding to \underline{Y} in the Bnet of Fig.30.1. Fig.30.2 shows a new Bnet that has a new node called $\vec{\underline{Y}}$ compared to the Bnet of Fig.30.1. One defines the TPMs for all nodes of Fig.30.2 except $\vec{\underline{Y}}$ and \underline{V} the same as for Fig.30.1. For $\vec{\underline{Y}}$ and \underline{V} , one defines

$$P(Y[\sigma] | \vec{\hat{y}}) = P(\underline{Y} = Y[\sigma] | \hat{y}[\sigma]) \quad (30.23)$$

$$P(V|\vec{\underline{Y}}, \vec{y}) = \delta(V, \frac{-1}{nsam(\vec{x})} \ln \mathcal{L}), \quad (30.24)$$

where $\mathcal{L} = \prod_{\sigma} P(\underline{Y} = y[\sigma] | \hat{y}[\sigma])$ =likelihood.

Chapter 31

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 \bar{G} called a DEN diagram by adding to each node a of G a non-deterministic root node \underline{u}_a pointing into a only. The nodes \underline{u}_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.

31.1 Example of LDEN diagram

The TPMs, printed in blue, for the nodes of the LDEN diagram Fig.31.1, are as follows.

$$P(y|w, z, \underline{u}_y) = \mathbb{1}(y = \epsilon w + \delta z + \underline{u}_y) \quad (31.1)$$

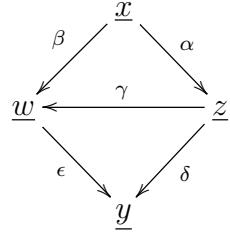


Figure 31.1: Example of a LDEN diagram wherein \underline{x} splits into two nodes \underline{z} and \underline{w} , then merges into node \underline{y} . There is also an arrow $\underline{z} \rightarrow \underline{w}$. Exogenous nodes are not shown. The Greek letters represent real numbers.

$$P(w|x, z, u_{\underline{w}}) = \mathbb{1}(w = \beta x + \gamma z + u_{\underline{w}}) \quad (31.2)$$

$$P(z|x, u_{\underline{z}}) = \mathbb{1}(z = \alpha x + u_{\underline{z}}) \quad (31.3)$$

$$P(x|u_{\underline{x}}) = \mathbb{1}(x = u_{\underline{x}}) \quad (31.4)$$

Hence,

$$y = \epsilon w + \delta z + u_{\underline{y}} \quad (31.5)$$

$$= \epsilon(\beta x + \gamma z + u_{\underline{w}}) + \delta z + u_{\underline{y}} \quad (31.6)$$

$$= (\epsilon\gamma + \delta)z + \epsilon\beta x + \epsilon u_{\underline{w}} + u_{\underline{y}} \quad (31.7)$$

$$= (\epsilon\gamma + \delta)z + \epsilon\beta u_{\underline{x}} + \epsilon u_{\underline{w}} + u_{\underline{y}} . \quad (31.8)$$

Therefore

$$\left(\frac{\partial y}{\partial z} \right)_{u_{\underline{w}}, u_{\underline{y}}} = \epsilon\gamma + \delta , \quad (31.9)$$

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.

31.2 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 Definition of a Bayesian Network. 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 Chapter Notational Conventions and Preliminaries.

Consider a LDEN diagram with deterministic nodes $\underline{x}_\cdot = (\underline{x}_k)_{k=0,1,\dots,nx-1}$ and corresponding exogenous nodes $\underline{u}_\cdot = (\underline{u}_k)_{k=0,1,\dots,nx-1}$. Assume $\langle \underline{u}_i, \underline{u}_j \rangle = 0$ if $i \neq j$. The strength of each connection $\underline{x}_i \rightarrow \underline{x}_j$ 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 $i \rightarrow j$ would not be drawn.

31.2.1 Fully connected LDEN diagram with $nx = 2$

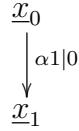


Figure 31.2: Fully connected LDEN diagram with two \underline{x}_j nodes (exogenous nodes \underline{u}_j not shown).

Consider the LDEN diagram of Fig.31.2. This diagram represents the following **structural equations**:

$$\underline{x}_0 = \underline{u}_0 \quad (31.10a)$$

$$\underline{x}_1 = \alpha_{1|0}\underline{x}_0 + \underline{u}_1. \quad (31.10b)$$

Eqs.31.10 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

$$\langle \underline{x}_1, \underline{x}_0 \rangle = \alpha_{1|0} \langle \underline{x}_0, \underline{x}_0 \rangle. \quad (31.11)$$

Thus, $\alpha_{1|0}$ can be estimated from the covariances $\langle \underline{x}_i, \underline{x}_j \rangle$.

31.2.2 Fully connected LDEN diagram with $nx = 3$

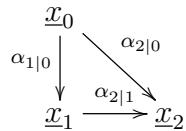


Figure 31.3: Fully connected LDEN diagram with three \underline{x}_j nodes (exogenous nodes \underline{u}_j not shown).

Consider the LDEN diagram of Fig.31.3. This diagram represents the following **structural equations**:

$$\underline{x}_0 = \underline{u}_0 \quad (31.12a)$$

$$\underline{x}_1 = \alpha_{1|0}\underline{x}_0 + \underline{u}_1 \quad (31.12b)$$

$$\underline{x}_2 = \alpha_{2|0}\underline{x}_0 + \alpha_{2|1}\underline{x}_1 + \underline{u}_2. \quad (31.12c)$$

Eqs.31.12 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

$$\langle \underline{x}_1, \underline{x}_0 \rangle = \alpha_{1|0} \langle \underline{x}_0, \underline{x}_0 \rangle \quad (31.13a)$$

$$\langle \underline{x}_2, \underline{x}_0 \rangle = \alpha_{2|0} \langle \underline{x}_0, \underline{x}_0 \rangle + \alpha_{2|1} \langle \underline{x}_1, \underline{x}_0 \rangle \quad (31.13b)$$

$$\langle \underline{x}_2, \underline{x}_1 \rangle = \alpha_{2|0} \langle \underline{x}_0, \underline{x}_1 \rangle + \alpha_{2|1} \langle \underline{x}_1, \underline{x}_1 \rangle \quad (31.13c)$$

Eqs.31.13 constitute a system of 3 linear equations in 3 unknowns (the α 's) so we can 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.

31.2.3 Fully connected LDEN diagram with arbitrary nx

Let $\underline{x}_. = (x_i)_{i=0,1,\dots,nx-1}$ and $\underline{x}_{<i} = (x_k)_{k=0,1,\dots,i-1}$. 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}_i of the LDEN diagram, be

$$P(x_i|\underline{x}_{<i}, \underline{u}_i) = \mathbb{1}(x_i = \sum_{k< i} \alpha_{i|k} x_k + u_i), \quad (31.14)$$

for some parameters $\alpha_{i|k} \in \mathbb{R}$. The exogenous nodes $\underline{u}_.$ are assumed to be independent so

$$P(\underline{u}_.) = \prod_i P(u_i) \quad (31.15)$$

and

$$\langle \underline{u}_i, \underline{u}_j \rangle = 0 \text{ if } i \neq j. \quad (31.16)$$

Note that

$$P(x_.) = \sum_{\underline{u}_.} P(\underline{u}_.) \prod_i P(x_i|\underline{x}_{<i}, u_i) \quad (31.17)$$

$$= E_{\underline{u}_.} [\prod_i P(x_i|\underline{x}_{<i}, u_i)]. \quad (31.18)$$

In terms of random variables, this system is described by the following **structural equations**:

$$\underline{x}_i = \sum_{k < i} \alpha_{i|k} \underline{x}_k + \underline{u}_i . \quad (31.19)$$

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 $nx = 4$,

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \alpha_{1|0} & 0 & 0 & 0 \\ \alpha_{2|0} & \alpha_{2|1} & 0 & 0 \\ \alpha_{3|0} & \alpha_{3|1} & \alpha_{3|2} & 0 \end{bmatrix} . \quad (31.20)$$

If we now represent the multinodes $\underline{x}_.$ and $\underline{u}_.$ as column vectors \underline{x} and \underline{u} , we get

$$\underline{x} = A \underline{x} + \underline{u} . \quad (31.21)$$

Note that

$$\underline{x} = (1 - A)^{-1} \underline{u} . \quad (31.22)$$

Therefore,

$$\underline{x}_i = f_i(\underline{u}_{\leq i}) . \quad (31.23)$$

Therefore, if $i > j$,

$$\langle \underline{u}_i, \underline{x}_j \rangle = \langle \underline{u}_i, f_j(\underline{u}_{\leq j}) \rangle = 0 . \quad (31.24)$$

Thus, if $i > j$,

$$\langle \underline{x}_i, \underline{x}_j \rangle = \sum_{k < i} \alpha_{i|k} \langle \underline{x}_k, \underline{x}_j \rangle + \langle \underline{u}_i, \underline{x}_j \rangle \quad (31.25)$$

$$= \sum_{k < i} \alpha_{i|k} \langle \underline{x}_k, \underline{x}_j \rangle . \quad (31.26)$$

In matrix notation, Eq.(31.26) becomes

$$\langle \underline{x}, \underline{x}^T \rangle_L = A[\langle \underline{x}, \underline{x}^T \rangle_L + \langle \underline{x}, \underline{x}^T \rangle_D] \quad (31.27)$$

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,

$$A = \langle \underline{x}, \underline{x}^T \rangle_L [\langle \underline{x}, \underline{x}^T \rangle_L + \langle \underline{x}, \underline{x}^T \rangle_D]^{-1} . \quad (31.28)$$

This gives an estimate for the α 's in terms of the covariances $\langle \underline{x}_i, \underline{x}_j \rangle$.

31.3 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.(31.14) for the TPMs of the deterministic nodes of the diagram by

$$P(x_i|x_{<i}, u_i) = \mathbb{1}(x_i = f_i(x_{<i}, u_i)) , \quad (31.29)$$

with structural equations

$$\underline{x}_i = f_i(\underline{x}_{<i}, \underline{u}_i) , \quad (31.30)$$

for $i = 0, 1, \dots, nx - 1$. 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

$$P(x_i|x_{<i}, u_i) = P(x_i) = \delta(x_i, a) \quad (31.31)$$

and

$$\underline{x}_i = a \quad (31.32)$$

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.(31.29) and (31.30) are the TPMs and structural equations for a fully connected, non-linear DEN diagram. For a non-fully connected diagram,

- replace the multinode $x_{<i}$ by a subset of itself, in Eqs.(31.29) and (31.30) , and
- delete the corresponding arrows from the graph.

Chapter 32

Markov Blankets

This chapter is based on the Wikipedia article, Ref.[74]. Markov blankets and Markov boundaries of bnets were apparently invented by Judea Pearl. His 1988 book Ref.[26], instead of a research paper, is usually given as the original reference.

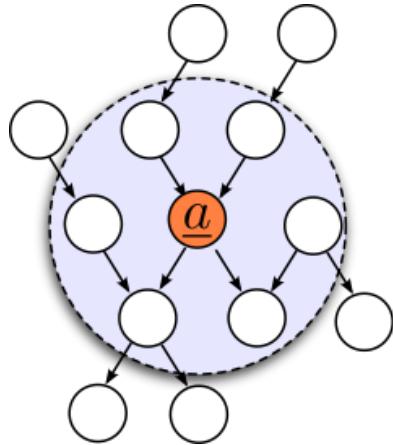


Figure 32.1: In a bnet, the minimal Markov blanket, aka Markov boundary, of node \underline{a} .

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

$$H(\underline{a} : \underline{b} | \underline{c}) = \sum_{a,b,c} P(a, b, c) \ln \frac{P(a, b | c)}{P(a | c)P(b | c)}. \quad (32.1)$$

$H(\underline{a} : \underline{b} | \underline{c}) = 0$ 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

$$H(\underline{a} : \underline{X} - \underline{a} | \underline{B}) = 0 . \quad (32.2)$$

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.32.1.

Chapter 33

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.[77]. 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 $P_{\underline{x}} : S_{\underline{x}} \rightarrow [0, 1]$ to denote the target probability distribution that we wish to obtain samples from.

33.1 Inverse Cumulative Sampling

For more info about this topic and some original references, see Ref.[64].

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:

$$CUM_{\underline{x}}(x) = P(\underline{x} < x) = \int_{x' < x} dx' P_{\underline{x}}(x') . \quad (33.1)$$

Note that

$$P_{\underline{x}}(x) = \frac{d}{dx} CUM_{\underline{x}}(x) . \quad (33.2)$$

For $t = 0, 1, \dots, T - 1$, let

$\underline{u}^{(t)} \in [0, 1]$ = random variable, uniformly distributed over $[0, 1]$.

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\dots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all σ . Vector of samples collected up to time t .

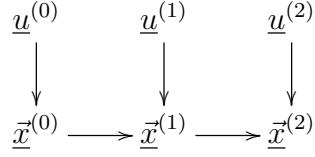


Figure 33.1: bnet for Inverse Cumulative Sampling

The TPMs, printed in blue, for the nodes of bnet Fig.33.1, are:

$$P(\underline{u}^{(t)}) = 1 \quad (33.3)$$

$$P(\underline{x}^{(t)} | \underline{x}^{(t-1)}, \underline{u}^{(t)}) = \delta(\underline{x}^{(t)}, [\underline{x}^{(t-1)}, CUM_{\underline{x}}^{-1}(\underline{u}^{(t)})]) \quad (33.4)$$

Motivation

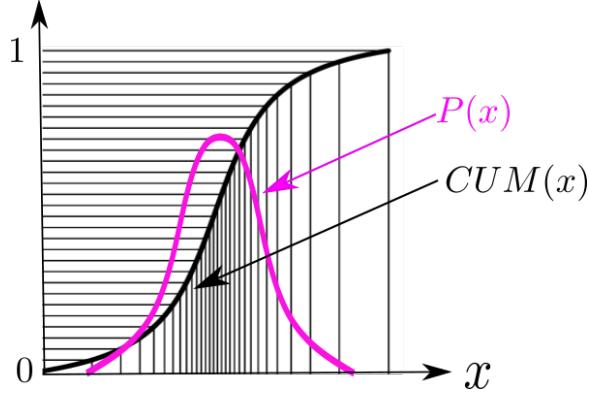


Figure 33.2: Motivation for Inverse Cumulative Sampling.

See Fig.33.2.

Note that if \underline{u} is uniformly distributed over the interval $[0, 1]$ and $a \in [0, 1]$, then

$$P(\underline{u} < a) = a . \quad (33.5)$$

Thus

$$P(CUM_{\underline{x}}^{-1}(\underline{u}) < x) = P(\underline{u} < CUM_{\underline{x}}(x)) \quad (33.6)$$

$$= CUM_{\underline{x}}(x) . \quad (33.7)$$

Therefore,

$$dP(CUM_{\underline{x}}^{-1}(\underline{u}) < x) = P_{\underline{x}}(x)dx . \quad (33.8)$$

33.2 Rejection Sampling

For more info about this topic and some original references, see Ref.[87].

This method samples from a “candidates” probability distribution $P_c : S_x \rightarrow [0, 1]$, in cases where sampling directly from the target probability distribution $P_{\underline{x}} : S_{\underline{x}} \rightarrow [0, 1]$ is not possible.

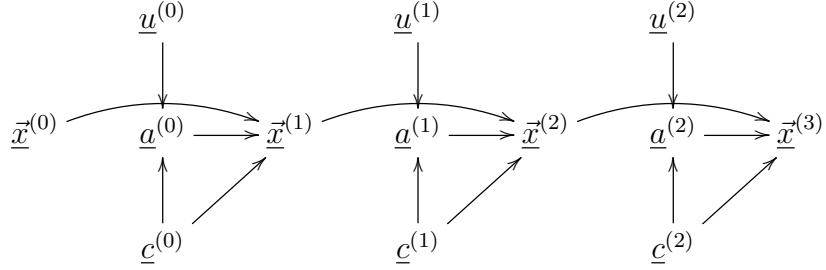


Figure 33.3: bnet for Rejection Sampling

For $t = 0, 1, \dots, T - 1$, 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)

$\underline{c}^{(t)} \in S_x$ = sample that is a candidate for being accepted

$\vec{x}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\dots,nsam(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_x$ for all σ . Vector of samples collected up to time t .

This algorithm requires a priori definition of a candidate probability distribution $P_c : S_x \rightarrow \mathbb{R}$ such that

$$P_{\underline{x}}(x) < \beta P_c(x) \quad (33.9)$$

for all $x \in S_x$, for some $\beta \in \mathbb{R}$.

The TPMs, printed in blue, for the nodes of bnet Fig.33.3, are:

$$P(u^{(t)} = u) = 1 \quad (33.10)$$

$$P(\underline{c}^{(t)} = c) = P_c(c) \quad (33.11)$$

$$P(\underline{a}^{(t)} = a | \underline{c}^{(t)} = c, \underline{u}^{(t)} = u) = \begin{cases} \delta(a, 0) & \text{if } u\beta P_c(c) \geq P_{\underline{x}}(c) \\ \delta(a, 1) & \text{if } u\beta P_c(c) < P_{\underline{x}}(c) \end{cases} \quad (33.12)$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{x}^{(t)}, \vec{x}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{x}^{(t)}, [\vec{x}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \quad (33.13)$$

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

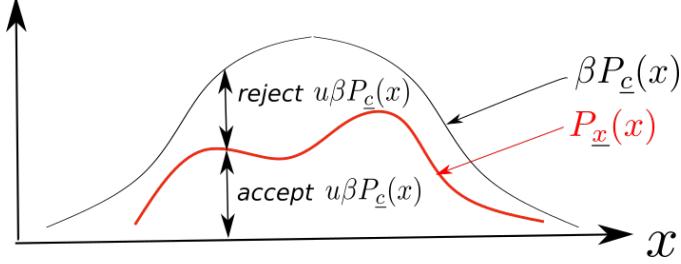


Figure 33.4: Motivation for Rejection Sampling.

See Fig.33.4.

33.3 Metropolis-Hastings Sampling

For more info about this topic and some original references, see Refs.[1] and [75].

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 $t = 0, 1, \dots, T - 1$, 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

$\vec{\underline{x}}^{(t)} = (\underline{x}^{(t)}[\sigma])_{\sigma=0,1,\dots,n_{sam}(t)-1}$ where $\underline{x}^{(t)}[\sigma] \in S_{\underline{x}}$ for all σ . Vector of samples collected up to time t .

A **proposal TPM** $P_{c|\underline{x}} : S_{\underline{x}}^2 \rightarrow [0, 1]$ must be specified a priori for this algorithm.

The TPMs, printed in blue, for the nodes of bnet Fig.33.5, are:

$$P(u^{(t)} = u) = 1 \quad (33.14)$$

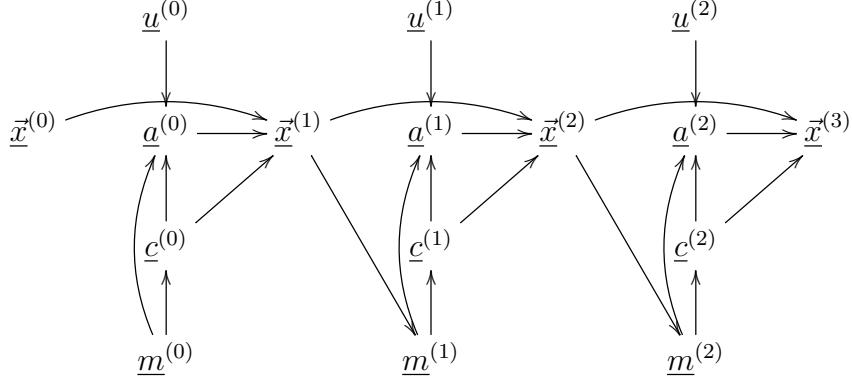


Figure 33.5: bnet for Metropolis-Hastings Sampling

$$P(\underline{c}^{(t)} = c | \underline{m}^{(t)} = m) = P_{\underline{c}|\underline{x}}(c|m) \quad (33.15)$$

$$P(\underline{a}^{(t)} = a | \underline{c}^{(t)} = c, \underline{u}^{(t)} = u, \underline{m}^{(t)} = m) = \begin{cases} \delta(a, 0) & \text{if } u \geq \alpha(c|m) \\ \delta(a, 1) & \text{if } u < \alpha(c|m) \end{cases} \quad (33.16)$$

where the **acceptance probability** α is defined as

$$\alpha(c|m) = \min \left(1, \frac{P_{\underline{c}|\underline{x}}(m|c) P_{\underline{x}}(c)}{P_{\underline{c}|\underline{x}}(c|m) P_{\underline{x}}(m)} \right). \quad (33.17)$$

Note that if the proposal distribution is symmetric, then

$$\alpha(c|m) = \min \left(1, \frac{P_{\underline{x}}(c)}{P_{\underline{x}}(m)} \right). \quad (33.18)$$

$$P(\vec{x}^{(t)} | \vec{x}^{(t-1)}, \underline{a}^{(t)} = a, \underline{c}^{(t)} = c) = \begin{cases} \delta(\vec{x}^{(t)}, \vec{x}^{(t-1)}) & \text{if } a = 0 \\ \delta(\vec{x}^{(t)}, [\vec{x}^{(t-1)}, c]) & \text{if } a = 1 \end{cases} \quad (33.19)$$

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.

$$P(\underline{m}^{(t)} = m | \vec{x}^{(t)}) = \delta(m, \text{last component of } \vec{x}^{(t)}). \quad (33.20)$$

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.33.6.

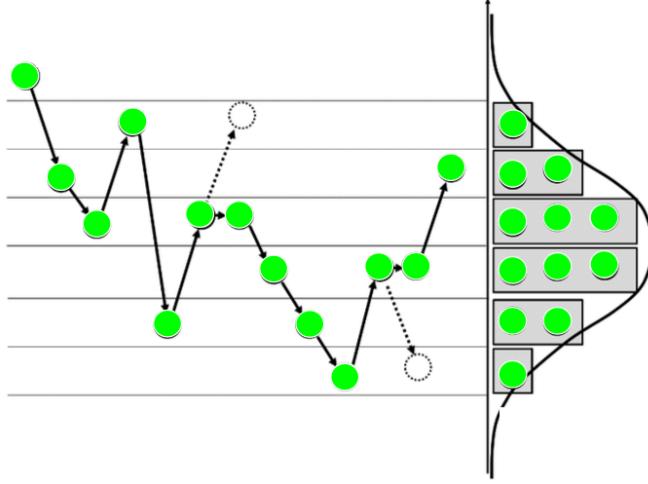


Figure 33.6: Motivation for Metropolis-Hastings Sampling.

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

$$\pi = \lim_{n \rightarrow \infty} T^n \pi_0 . \quad (33.21)$$

Suppose the prob distribution $P_{\underline{x}}(x)$ that we wish to sample from satisfies

$$P_{\underline{x}}(x) = \pi(x) . \quad (33.22)$$

Reversibility (detailed balance): For all $x, x' \in S_{\underline{x}}$,

$$P(x'|x)\pi(x) = P(x|x')\pi(x') . \quad (33.23)$$

Detailed balance is a sufficient (although not necessary) condition for a unique stationary prob distribution π to exist.¹

Let

$$P(x'|x) = P(\underline{a} = 1|x', x)P_{\underline{c}|x}(x'|x) + \delta(x, x')P(\underline{a} = 0|x) , \quad (33.24)$$

where

$$P(\underline{a} = 0|x) = \sum_{x'} P(\underline{a} = 0|x', x)P_{\underline{c}|x}(x'|x) . \quad (33.25)$$

Claim 23 *If*

$$P(\underline{a} = 1|x', x) = \alpha(x'|x) , \quad (33.26)$$

¹ As explained lucidly in Ref.[1], besides detailed balance, 2 other properties must also be satisfied by the Markov chain, irreducibility 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 $x \neq x'$.

$$P(x'|x)P(x) = P(\underline{a} = 1|x', x)P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) \quad (33.27)$$

$$= \min \left(1, \frac{P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')}{P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x)} \right) P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x) \quad (33.28)$$

$$= \min (P_{\underline{c}|\underline{x}}(x'|x)P_{\underline{x}}(x), P_{\underline{c}|\underline{x}}(x|x')P_{\underline{x}}(x')) \quad (33.29)$$

$$= P(x|x')P(x') \quad (33.30)$$

QED

33.4 Gibbs Sampling

For more info about this topic and some original references, see Ref.[60].

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 $\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{N-1}$

Identify the random variable $\underline{x} = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{N-1})$ with the random variable \underline{x} used in Metropolis-Hastings sampling. For Gibbs sampling, we use the following proposal distribution:

$$P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j | [m_i]_{i \neq j}) . \quad (33.31)$$

Eq.(33.31) can be simplified using Markov Blankets (see Chapter 32) to the following:

$$P_{\underline{c}|\underline{x}}(c|m) = \prod_{j=0}^{N-1} P(c_j | [m_i : \forall i \ni \underline{x}_i \in MB(\underline{x}_j)]) , \quad (33.32)$$

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.33.7. The TPM for the nodes of that bnet are

$$P(\underline{c}_j^{(t)} = c_j | (\underline{c}_i^{(t)})_{i < j} = (c_i)_{i < j}, \underline{m}^{(t-1)} = m) = P(c_j | (c_i)_{i < j}, (m_i)_{i > j}) \quad (33.33)$$

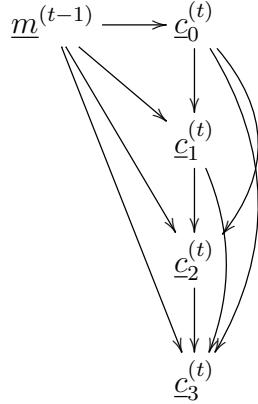


Figure 33.7: In Gibbs sampling, the proposal distribution $P_{\underline{c}|\underline{x}}$ can be defined by making the components $c_j^{(t)}$ of candidate sample $\underline{c}^{(t)}$ depend on the previous components $(c_i^{(t)})_{i < j}$.

for $j = 0, 1, \dots, N - 1$. This implies

$$P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c | \underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j | (c_i)_{i < j}, (m_i)_{i > j}) . \quad (33.34)$$

As before, we can condition only on the Markov blanket of each node \underline{x}_j .

$$P_{\underline{c}|\underline{x}}(\underline{c}^{(t)} = c | \underline{m}^{(t-1)} = m) = \prod_{j=0}^{N-1} P(c_j | (c_i)_{i < j}, (m_i)_{i > j}, \text{ use only } c_i \text{ and } m_i \ni \underline{x}_i \in MB(\underline{x}_j)) . \quad (33.35)$$

33.5 Importance Sampling

For more info about this topic and some original references, see Ref.[62].

Suppose random variables $\underline{x}[\sigma] \in S_{\underline{x}}$ for $\sigma = 0, 1, \dots, nsam - 1$ are i.i.d. with probability distribution $P_{\underline{x}}$. Then

$$E_{\underline{x}}[f(x)] \approx \frac{1}{nsam} \sum_{\sigma=0}^{nsam-1} f(x[\sigma]) \quad (33.36)$$

for any $f : S_{\underline{x}} \rightarrow \mathbb{R}$. 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 $\underline{y}[\sigma] \sim P_{\underline{y}}$.

$$E_{\underline{x}}[f(\underline{x})] = \sum_x P_{\underline{x}}(x) f(x) \quad (33.37)$$

$$= \sum_x P_{\underline{y}}(x) \frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} f(x) \quad (33.38)$$

$$= E_{\underline{y}}\left[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)\right] \quad (33.39)$$

Sampling from $P_{\underline{y}}(y)$ instead of $P_{\underline{x}}(x)$ might reduce (or increase) variance for a particular $f : S_{\underline{x}} \rightarrow \mathbb{R}$.

$$Var_{\underline{x}}[f(x)] = E_{\underline{x}}[(f(x))^2] - (E_{\underline{x}}[f(x)])^2 \quad (33.40)$$

$$Var_{\underline{y}}\left[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)\right] = E_{\underline{y}}\left[\left(\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)\right)^2\right] - (E_{\underline{y}}\left[\frac{P_{\underline{x}}(y)}{P_{\underline{y}}(y)} f(y)\right])^2 \quad (33.41)$$

$$= E_{\underline{x}}\left[\frac{P_{\underline{x}}(x)}{P_{\underline{y}}(x)} (f(x))^2\right] - (E_{\underline{x}}[f(x)])^2 \quad (33.42)$$

Chapter 34

Markov Chains

A Markov Chain is simply a bnet with the graph structure of a chain. For example, Fig.34.1 shows a chain with $n = 4$ nodes.

$$\underline{x}_0 \longrightarrow \underline{x}_1 \longrightarrow \underline{x}_2 \longrightarrow \underline{x}_3$$

Figure 34.1: 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:

$$P(x_t | (x_a)_{a \neq t}) = P(x_t | x_{t-1}), \quad (34.1)$$

where $(x_a)_{a \neq t}$ are all the nodes except x_t itself and $t = 1, 2, \dots, n - 1$.

If there exists a single TPM $P_{\underline{x}_1 | \underline{x}_0}$ such that

$$P(x_t | x_{t-1}) = P_{\underline{x}_1 | \underline{x}_0}(x_t | x_{t-1}) \quad (34.2)$$

for $t = 1, 2, \dots, n - 1$, then we say that the Markov chain is **time homogeneous**.

Claim 24 (*Data Processing Inequality (DPI)*)

Consider a Markov chain $\underline{x}_0 \rightarrow \underline{x}_1 \cdots \rightarrow \underline{x}_{n-1}$. Suppose $0 \leq a < m < b \leq n - 1$. Then

$$H(\underline{x}_a : \underline{x}_b) \leq \min[H(\underline{x}_a : \underline{x}_m), H(\underline{x}_m : \underline{x}_b)] \quad (34.3)$$

See Ref.[54] for references where the DPI is proven. This inequality confirms our intuitive expectations that the information transmitted (i.e., the mutual information(MI)) from a to b (or vice versa since MI is symmetric) is smaller or equal to the one transmitted from a to m or from m to b because a and b are “farther apart” and “some info can get lost during transmission through the mediator node m ”.

Chapter 35

Message Passing (Belief Propagation)

Belief Propagation was first proposed in 1982 Ref.[25] 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 28) 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.[26], 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 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.[26] by Pearl. Refs.[45], and [19] were also helpful in writing this chapter.

35.1 Distributed Soldier Counting

Consider a group of soldiers marching single file. Fig.35.1 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.

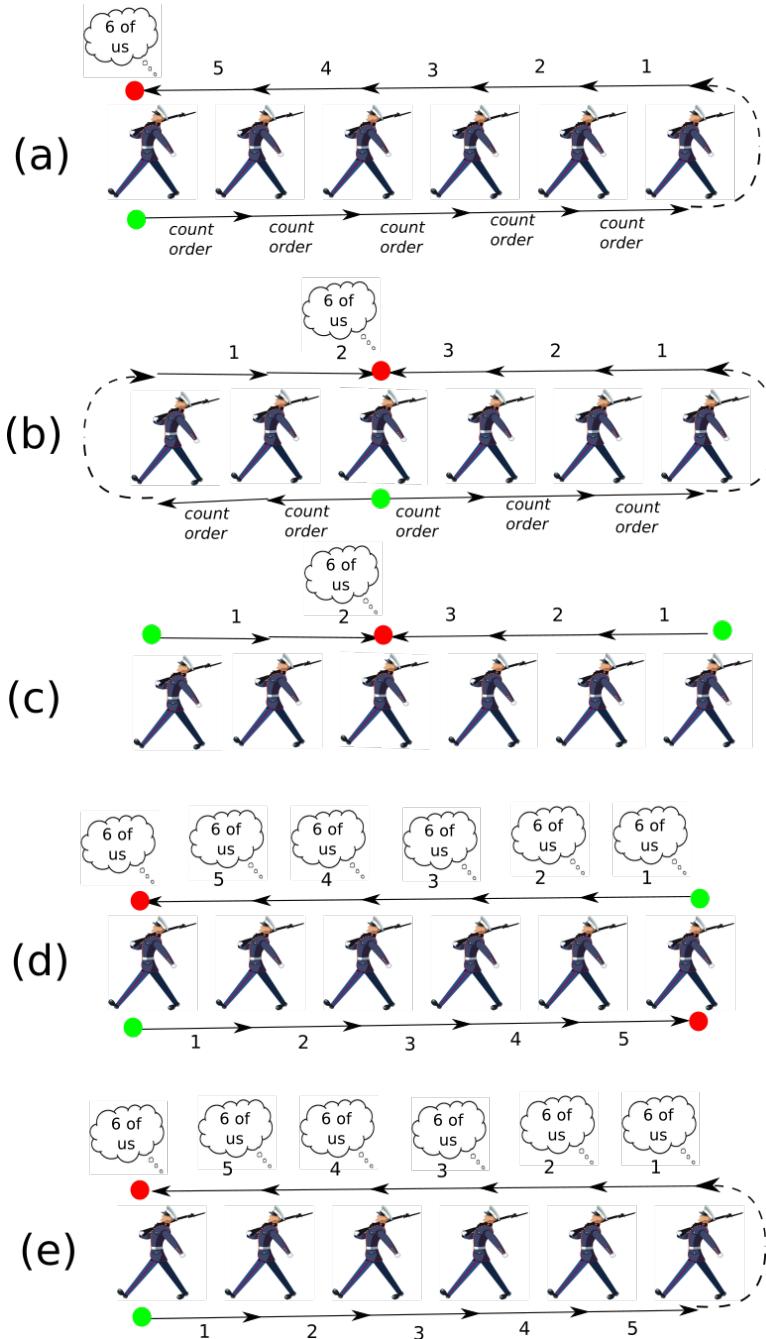


Figure 35.1: Distributed soldier counting (This example comes from Chapter 4 of Ref.[26]). Green dots indicate the beginning and red dots the end of a count. Only first soldier can calculate total count in (a). Only third soldier can calculate total count in (b,c). All soldiers can calculate the total count in (d,e). One starting point in (a,b,e). Two ends as starting points in (c,d).

35.2 Spring Systems

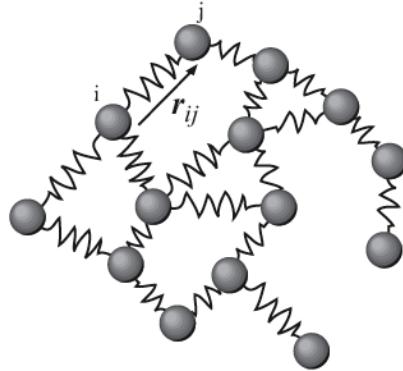


Figure 35.2: Spring system. Point masses connected by springs.

See Ref.[90] for an introduction to spring systems. Ideal springs between the point mass nodes would not be sufficient. One would have to add damping to the springs so as to reach an equilibrium. Time dependent forces (loads) pointing into or out of the page, applied to the point masses, would generate signals that would propagate like BP messages.

35.3 BP for Markov Chains

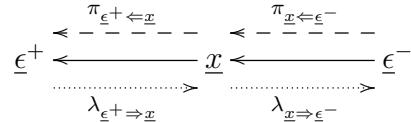


Figure 35.3: 3 node Markov chain $\underline{\epsilon}^+ \leftarrow x \leftarrow \underline{\epsilon}^-$. The π messages (probability functions) travel downstream (i.e., they carry info in the direction of the graph arrows, towards the future) and are indicated by a dashed arrow or by a left double arrow \Leftarrow . The λ messages (likelihood functions) travel upstream (i.e., they carry info opposite to direction of the graph arrows, towards the past) and are indicated by a dotted arrow or by a right double arrow \Rightarrow . $\underline{\epsilon}^+$ 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.35.3. Define¹

$$\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x) = P(x|\epsilon^-) \text{ (past of } \underline{x}) \quad (35.1)$$

$$\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x) = P(\epsilon^+|x) \text{ (future of } \underline{x}) \quad (35.2)$$

$$\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^-) = \delta(\epsilon^-, \epsilon_0^-) \text{ (past of } \underline{\epsilon}^-) \quad (35.3)$$

$$\lambda_{\underline{x} \Rightarrow \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^+|\epsilon^-) \text{ (future of } \underline{\epsilon}^-) . \quad (35.4)$$

Furthermore, define the Belief BEL in x to be

$$BEL_{\underline{x}}(x) = P(x|\epsilon) , \quad (35.5)$$

where

$$\epsilon = \underline{\epsilon}^+ \cup \underline{\epsilon}^- . \quad (35.6)$$

It follows that

$$BEL_{\underline{x}}(x) = P(x|\epsilon^+, \epsilon^-) = \quad (35.7)$$

$$= \mathcal{N}(!x)P(\epsilon^+, x, \epsilon^-) \quad (35.8)$$

$$= \mathcal{N}(!x)P(\epsilon^+|x)P(x|\epsilon^-) \quad (35.9)$$

$$= \mathcal{N}(!x)\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x)\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x) . \quad (35.10)$$

Note that Bayes rule would affirm that²

$$P(x|\epsilon^+) = \mathcal{N}(!x) \underbrace{P(\epsilon^+|x)}_{\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x)} P(x) . \quad (35.11)$$

Thus, Eq.(35.10) is like a 2-sided Janus Bayes rule.

Note that the π messages and λ messages propagate independently of each other, via the TPM $P(x|\epsilon^-)$:

$$\underbrace{\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x)}_{P(x|\epsilon_0^-)} = \sum_{\epsilon^-} P(x|\epsilon^-) \underbrace{\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-)}_{\delta(\epsilon^-, \epsilon_0^-)} \quad (35.12a)$$

¹The pattern behind these definitions, in case it eludes you, is as follows: the π 's always carry information about the past and the λ 's about the future. But the past or future of what? Of the argument of the function. Out of the two random variables in the subscript of the function, the one on the right hand side of the subscript, the one which is adjacent but beneath the argument, is always the argument.

²As usual in this book, $\mathcal{N}(!x)$ means a constant that is independent of x .

$$\underbrace{\lambda_{\underline{x} \Rightarrow \underline{\epsilon}^-}(\epsilon^-)}_{P(\epsilon^+ | \epsilon^-)} = \sum_x P(x | \epsilon^-) \underbrace{\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x)}_{P(\epsilon^+ | x)} \quad (35.12b)$$

Eqs.(35.12) 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.35.4, is complementary to bnet for the nodes themselves. We will call it the **BP 2-track bnet** for the bnet Fig.35.3, 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.35.4, are as follows:

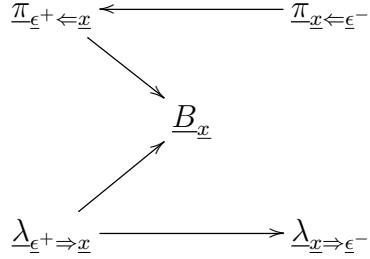


Figure 35.4: BP 2-track bnet for the bnet Fig.35.3.

$$P(\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}) = \prod_{\epsilon^-} \mathbb{1}(\pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-) = P(\epsilon^-)) \quad (35.13)$$

$$P(\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}} | \pi_{\underline{x} \leftarrow \underline{\epsilon}^-}) = \prod_x \mathbb{1} \left(\pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}(x) = \sum_{\epsilon^-} P(x | \epsilon^-) \pi_{\underline{x} \leftarrow \underline{\epsilon}^-}(\epsilon^-) \right) \quad (35.14)$$

$$P(B_{\underline{x}} | \pi_{\underline{\epsilon}^+ \leftarrow \underline{x}}, \lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}) = \prod_x \mathbb{1} (B_{\underline{x}}(x) = BEL_{\underline{x}}(x)) \quad (35.15)$$

$$P(\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}) = \prod_x \mathbb{1} (\lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x) = P(\epsilon^+ | x)) \quad (35.16)$$

$$P(\lambda_{\underline{x} \Rightarrow \underline{\epsilon}^-} | \lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}) = \prod_{\epsilon^-} \mathbb{1} \left(\lambda_{\underline{x} \Rightarrow \underline{\epsilon}^-}(\epsilon^-) = \sum_x P(x | \epsilon^-) \lambda_{\underline{\epsilon}^+ \Rightarrow \underline{x}}(x) \right) \quad (35.17)$$

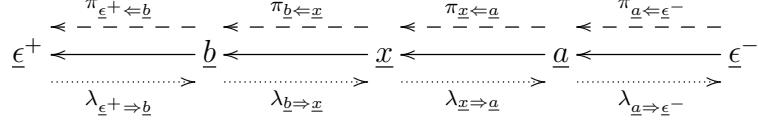


Figure 35.5: 5 node Markov chain

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.35.5. 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

$$\pi_{\underline{b} \leftarrow \underline{x}}(x) = P(x|\underline{\epsilon}^-) \text{ (past of } \underline{x}) , \quad (35.18)$$

$$\lambda_{\underline{b} \Rightarrow \underline{x}}(x) = P(\underline{\epsilon}^+|x) \text{ (future of } \underline{x}) , \quad (35.19)$$

$$\pi_{\underline{x} \leftarrow \underline{a}}(a) = P(a|\underline{\epsilon}^-) \text{ (past of } \underline{a}) \quad (35.20)$$

and

$$\lambda_{\underline{x} \Rightarrow \underline{a}}(a) = P(\underline{\epsilon}^+|a) \text{ (future of } \underline{a}) . \quad (35.21)$$

Define the Belief BEL in x to be

$$BEL_{\underline{x}}(x) = P(x|\underline{\epsilon}) , \quad (35.22)$$

where

$$\underline{\epsilon} = \underline{\epsilon}^+ \cup \underline{\epsilon}^- . \quad (35.23)$$

Then

$$BEL_{\underline{x}}(x) = \mathcal{N}(!x)P(\underline{\epsilon}^+|x)P(x|\underline{\epsilon}^-) \quad (35.24)$$

$$= \mathcal{N}(!x)\lambda_{\underline{b} \Rightarrow \underline{x}}(x)\pi_{\underline{b} \leftarrow \underline{x}}(x) . \quad (35.25)$$

In analogy with the case of BP for a 3 node Markov chain, we can define the bnet Fig.35.6, which we refer to as the **BP 2-track bnet** for Fig.35.5. The TPMs, printed in blue, for the nodes of bnet Fig.35.6, are as follows:

$$P(\pi_{\underline{b} \leftarrow \underline{x}}|\pi_{\underline{x} \leftarrow \underline{a}}) = \prod_x \mathbb{1} \left(\pi_{\underline{b} \leftarrow \underline{x}}(x) = \sum_a P(x|a)\pi_{\underline{x} \leftarrow \underline{a}}(a) \right) \quad (35.26)$$

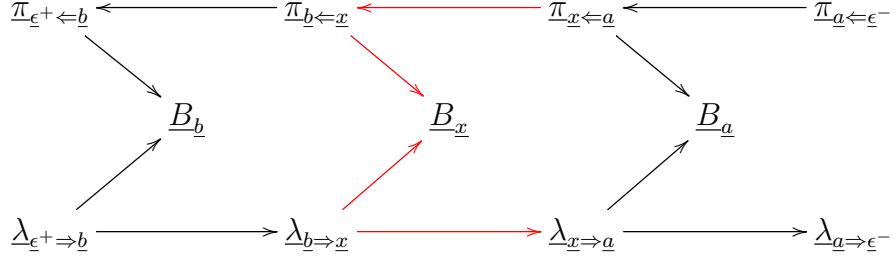


Figure 35.6: BP 2-track bnet for the bnet Fig.35.5.

$$P(B_{\underline{x}} | \pi_{\underline{b} \leftarrow \underline{x}}, \lambda_{\underline{b} \Rightarrow \underline{x}}) = \prod_x \mathbb{1}(B_{\underline{x}}(x) = BEL_{\underline{x}}(x)) \quad (35.27)$$

$$P(\lambda_{\underline{x} \Rightarrow \underline{a}} | \lambda_{\underline{b} \Rightarrow \underline{x}}) = \prod_a \mathbb{1} \left(\lambda_{\underline{x} \Rightarrow \underline{a}}(a) = \sum_x P(x|a) \lambda_{\underline{b} \Rightarrow \underline{x}}(x) \right) \quad (35.28)$$

Let us represent the Markov chain of Fig.35.5 by $\underline{x}_{nx-1} \leftarrow \dots \leftarrow \underline{x}_2 \leftarrow \underline{x}_1 \leftarrow \underline{x}_0$ where $nx = 5$. For any node \underline{x}_i with parent $\underline{p}_{\underline{x}_i} = \underline{x}_{i-1}$ and child $\underline{c}_{\underline{x}_i} = \underline{x}_{i+1}$, define the **memory matrix** $\mathcal{M}_{\underline{x}_i}$ for node \underline{x}_i as

$$\mathcal{M}_{\underline{x}_i} = [\mathcal{M}_{\underline{x}_i}^+, \mathcal{M}_{\underline{x}_i}^-], \quad (35.29)$$

where $+$ =future, $-$ =past, and

$$\mathcal{M}_{\underline{x}_i}^+ = \begin{bmatrix} \pi_{\underline{c}_{\underline{x}_i} \leftarrow \underline{x}_i}(\cdot) \\ \lambda_{\underline{c}_{\underline{x}_i} \Rightarrow \underline{x}_i}(\cdot) \end{bmatrix}, \quad \mathcal{M}_{\underline{x}_i}^- = \begin{bmatrix} \pi_{\underline{x}_i \leftarrow \underline{p}_{\underline{x}_i}}(\cdot) \\ \lambda_{\underline{x}_i \Rightarrow \underline{p}_{\underline{x}_i}}(\cdot) \end{bmatrix}. \quad (35.30)$$

Note that

$$\mathcal{M}_{\underline{x}_i}^- = \mathcal{M}_{\underline{p}_{\underline{x}_i}}^+ \quad (35.31)$$

for all nodes \underline{x}_i . We will refer to Eqs.(35.31) as the **memory overlap conditions**.

We will also use a permuted version of the memory matrix

$$\mathcal{M}'_{\underline{x}_i} = [\mathcal{M}_{\underline{x}_i}^{OUT}, \mathcal{M}_{\underline{x}_i}^{IN}], \quad (35.32)$$

where

$$\mathcal{M}_{\underline{x}_i}^{OUT} = \begin{bmatrix} \pi_{\underline{c}_{\underline{x}_i} \leftarrow \underline{x}_i}(\cdot) \\ \lambda_{\underline{x}_i \Rightarrow \underline{p}_{\underline{x}_i}}(\cdot) \end{bmatrix}, \quad \mathcal{M}_{\underline{x}_i}^{IN} = \begin{bmatrix} \pi_{\underline{x}_i \leftarrow \underline{p}_{\underline{x}_i}}(\cdot) \\ \lambda_{\underline{c}_{\underline{x}_i} \Rightarrow \underline{x}_i}(\cdot) \end{bmatrix}. \quad (35.33)$$

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

$$\underline{\mathcal{M}}_{\epsilon^+} \longleftarrow \underline{\mathcal{M}}_b \longleftarrow \underline{\mathcal{M}}_x \longleftarrow \underline{\mathcal{M}}_a \longleftarrow \underline{\mathcal{M}}_{\epsilon^-}$$

Figure 35.7: BP Memory Bnet for the bnet Fig.35.5.

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 the DAG but replaces each node \underline{x}_i by a memory $\underline{\mathcal{M}}_{\underline{x}_i}$. Fig.35.7 shows the memory bnet for the bnet Fig.35.5. The TPM, printed in blue, for the node $\underline{\mathcal{M}}_x$ of the memory bnet Fig.35.7, is as follows

$$P(\underline{\mathcal{M}}_{\underline{x}_i} | \underline{\mathcal{M}}_{n \in nb(\underline{x}_i)}) = AB , \quad (35.34)$$

where

$$A = \mathbb{1}(\underline{\mathcal{M}}_{\underline{x}_i}^- = \underline{\mathcal{M}}_{px_i}^+) , \quad (35.35)$$

and

$$B = \mathbb{1}(\underline{\mathcal{M}}_{\underline{x}_i}^{OUT} = \mathcal{C}(\underline{\mathcal{M}}_{\underline{x}_i}^{IN})) . \quad (35.36)$$

The function \mathcal{C} , which we will call the **BP local computation**, maps $\underline{\mathcal{M}}_{\underline{x}_i}^{IN}$ into $\underline{\mathcal{M}}_{\underline{x}_i}^{OUT}$. More explicitly, \mathcal{C} is defined so that

$$B = \underbrace{P(\pi_{b \Leftarrow x} | \pi_{x \Leftarrow a})}_{B_\pi} \underbrace{P(\lambda_{x \Rightarrow a} | \lambda_{b \Rightarrow x})}_{B_\lambda} , \quad (35.37)$$

where B_π and B_λ are given by Eqs.(35.26) and (35.28), respectively.

The BP memory bnet Fig. 35.7 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.³ To solve the CSE represented by the memory bnet Fig.35.7, we will use the dynamic bnet Fig.35.8. Henceforth, we will refer to Fig.35.8 as an **BP dynamic bnet** for Fig.35.7.

Next, we will explain the meaning of Fig.35.8. Fig.35.8 is a step by step recipe (i.e., algorithm) for solving a SCE, where the unknowns are memory matrices. Each step encoded in Fig.35.8 corresponds to a specific message sending event, where the messages are sent along the edges of the Markov chain Fig.35.5. These message sending events are portrayed in chronological order in Fig.35.9. In that figure, π

³ The term dynamic bnet was used in Chapter 17 to mean a time inhomogeneous Markov chain, but here we are stretching its meaning to include Markov chains that aren't time inhomogeneous.

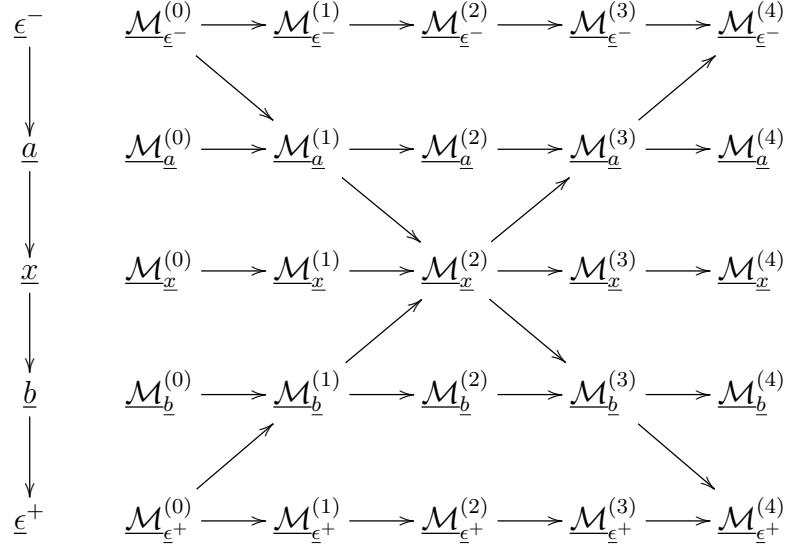


Figure 35.8: BP dynamic bnet for the bnet Fig.35.7.

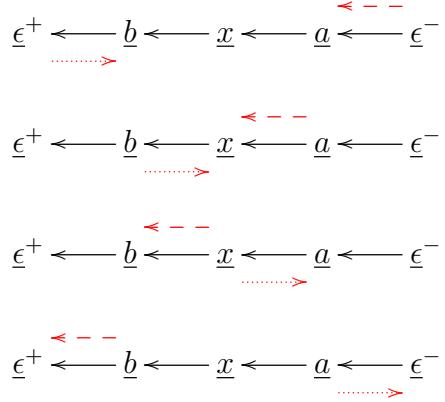


Figure 35.9: Steps encoded in the bnet Fig.35.8. Note the similarity of this figure to Fig.35.1 (d) for soldier counting.

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 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.35.8, 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

$$\pi_{\underline{p}\underline{x}_0 \leftarrow x_0}^{(0)} = P(x_0) , \quad (35.38)$$

$$\lambda_{\underline{cx}_i \Rightarrow \underline{x}_{nx-1}}^{(0)}(x_{nx-1}) = \delta(x_{nx-1}, x'_{nx-1}) . \quad (35.39)$$

All other $\mathcal{M}_{\underline{x}_i}^{(0)}$ entries for all i can be set to 1.

The TPMs, printed in blue, for the nodes of Fig.35.8, can all be summarized by

$$P(\mathcal{M}_{\underline{x}_i}^{(t)} | \mathcal{M}_{\underline{n} \in nb(\underline{x}_i)}^{(t-1)}, \mathcal{M}_{\underline{x}_i}^{(t-1)}) = AB , \quad (35.40)$$

where

$$A = \begin{cases} \mathbb{1}(\mathcal{M}_{\underline{x}_i}^{(t)-} = \mathcal{M}_{\underline{px}_i}^{(t-1)+}) & \text{if input from } \underline{px}_i \\ \mathbb{1}(\mathcal{M}_{\underline{x}_i}^{(t)+} = \mathcal{M}_{\underline{cx}_i}^{(t-1)-}) & \text{if input from } \underline{cx}_i \end{cases} , \quad (35.41)$$

and

$$B = \mathbb{1}(\mathcal{M}_{\underline{x}_i}^{(t)OUT} = \mathcal{C}(\mathcal{M}_{\underline{x}_i}^{(t)IN})) . \quad (35.42)$$

The function \mathcal{C} , which we will call the **BP local computation**, maps $\mathcal{M}_{\underline{x}_i}^{(t)IN}$ into $\mathcal{M}_{\underline{x}_i}^{(t)OUT}$. More explicitly, \mathcal{C} is defined so that

$$B = B_\pi B_\lambda \quad (35.43)$$

where

$$B_\pi = \prod_x \mathbb{1} \left(\underbrace{\pi_{\underline{b} \Leftarrow \underline{x}}^{(t)}(x)}_{OUT} = \sum_a P(x|a) \underbrace{\pi_{\underline{x} \Leftarrow \underline{a}}^{(t)}(a)}_{IN} \right) \quad (35.44)$$

and

$$B_\lambda = \prod_a \mathbb{1} \left(\underbrace{\lambda_{\underline{x} \Rightarrow \underline{a}}^{(t)}(a)}_{OUT} = \sum_x P(x|a) \underbrace{\lambda_{\underline{b} \Rightarrow \underline{x}}^{(t)}(x)}_{IN} \right) . \quad (35.45)$$

The basic idea behind Eq.(35.42), 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 .

35.4 BP Algorithm for Polytrees

Consider Fig.35.10, which illustrates a bnet node \underline{x} receiving and sending messages to its neighbors. The π messages (probability functions) travel downstream (i.e.,

they carry info in the direction of the graph arrows, towards the future) and are indicated by a dashed arrow or by a left double arrow \Leftarrow . The λ messages (likelihood functions) travel upstream (i.e., they carry info opposite to direction of the graph arrows, towards the past) and are indicated by a dotted arrow or by a right double arrow \Rightarrow .

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.35.10, 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}

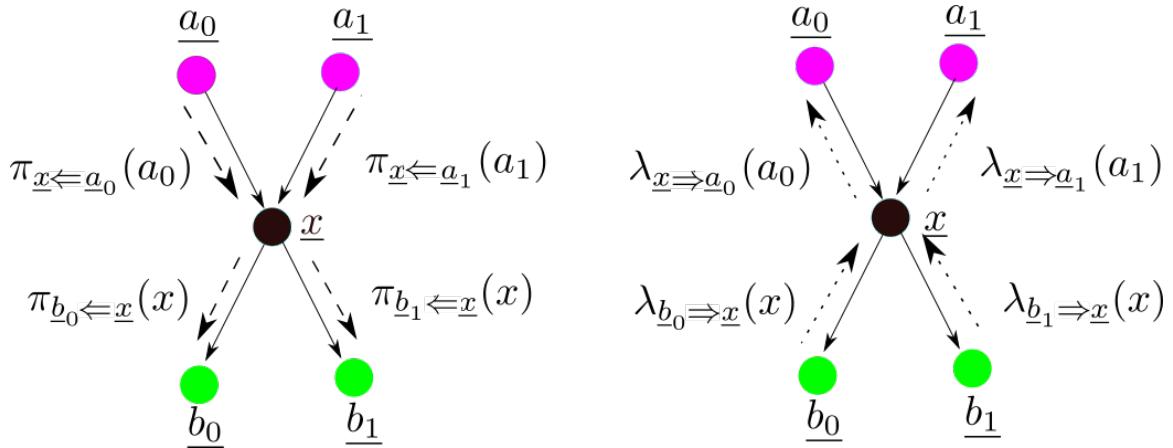


Figure 35.10: Node \underline{x} receiving and sending messages to its neighbors. (neighbors=parents and children).

We define a **memory matrix** $\mathcal{M}_{\underline{x}}$ for node \underline{x} as

$$\mathcal{M}_{\underline{x}} = [\mathcal{M}_{\underline{x}}^+, \mathcal{M}_{\underline{x}}^-] , \quad (35.46)$$

where $+$ =future, $-$ =past, and

$$\mathcal{M}_{\underline{x}}^+ = [\pi_{b \Leftarrow \underline{x}}(\cdot) \quad \lambda_{b \Rightarrow \underline{x}}(\cdot)]_{b \in ch(\underline{x})} = [\mathcal{M}_{b,\underline{x}}^+]_{b \in ch(\underline{x})} , \quad (35.47)$$

$$\mathcal{M}_{\underline{x}}^- = \left[\begin{array}{c} \pi_{\underline{x} \Leftarrow a}(\cdot) \\ \lambda_{\underline{x} \Rightarrow a}(\cdot) \end{array} \right]_{a \in pa(\underline{x})} = [\mathcal{M}_{\underline{x},a}^-]_{a \in pa(\underline{x})} . \quad (35.48)$$

Note that

$$\mathcal{M}_{\underline{x},a}^- = \mathcal{M}_{a,\underline{x}}^+ \quad (35.49)$$

for every arrow $x \leftarrow \underline{a}$. We will refer to Eqs.(35.49) as the **memory overlap conditions**.

We will also use a permuted version of the memory matrix

$$\mathcal{M}'_{\underline{x}} = [\mathcal{M}_{\underline{x}}^{OUT}, \mathcal{M}_{\underline{x}}^{IN}] , \quad (35.50)$$

where

$$\mathcal{M}_{\underline{x}}^{OUT} = \left(\begin{array}{c} [\pi_{b \leftarrow \underline{x}}(\cdot)]_{b \in ch(\underline{x})} \\ [\lambda_{\underline{x} \Rightarrow \underline{a}}(\cdot)]_{\underline{a} \in pa(\underline{x})} , \end{array} \right) = [\mathcal{M}_{\underline{x}, \underline{n}}^{OUT}]_{\underline{n} \in nb(\underline{x})} , \quad (35.51)$$

$$\mathcal{M}_{\underline{x}}^{IN} = \left(\begin{array}{c} [\pi_{\underline{x} \leftarrow \underline{a}}(\cdot)]_{\underline{a} \in pa(\underline{x})} \\ [\lambda_{\underline{b} \Rightarrow \underline{x}}(\cdot)]_{\underline{b} \in ch(\underline{x})} \end{array} \right) = [\mathcal{M}_{\underline{x}, \underline{n}}^{IN}]_{\underline{n} \in nb(\underline{x})} . \quad (35.52)$$

For times $t = 0, 1, \dots, T - 1$, we calculate $\mathcal{M}_{\underline{x}}^{(t)}$ in two steps: first we calculate $\mathcal{M}_{\underline{x}}^{(t)IN}$ from earlier memories at time $t - 1$, then we calculate $\mathcal{M}_{\underline{x}}^{(t)OUT}$:

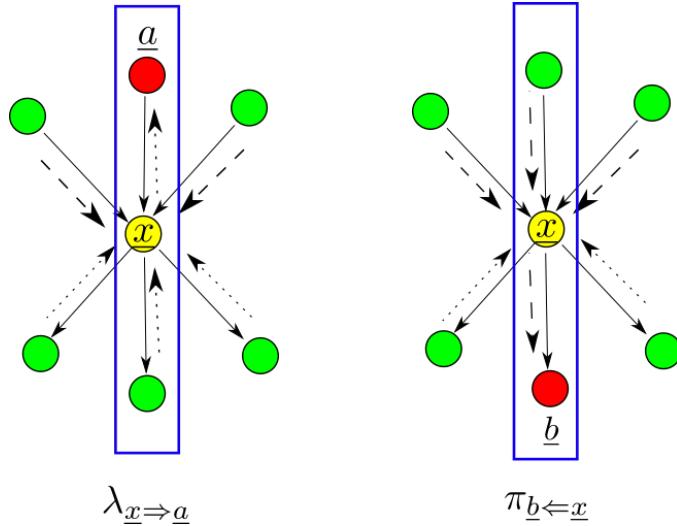


Figure 35.11: Subgraph of a bnet showing two cases (RULE 1 and RULE 2) of message info flow. The yellow node is a gossip monger. It receives messages from all the green nodes, and then it relays a joint message to the red node. Union of green nodes and the red node = full neighborhood of yellow node. 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 $\mathcal{M}_{\underline{x}}^{(t)IN}$ from signals received from $\underline{n} \in nb(\underline{x})$, sent at earlier time $t - 1$:

Set

$$\mathcal{M}_{\underline{x}, \underline{a}}^{(t)-} |_{\pi} = \mathcal{M}_{\underline{a}, \underline{x}}^{(t-1)+} |_{\pi} , \quad (35.53)$$

for all $\underline{a} \in pa(\underline{x})$, and

$$\mathcal{M}_{\underline{b}, \underline{x}}^{(t)+} |_{\lambda} = \mathcal{M}_{\underline{x}, \underline{b}}^{(t-1)-} |_{\lambda} , \quad (35.54)$$

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 $\mathcal{M}_{\underline{x}}^{(t)OUT}$ from already calculated $\mathcal{M}_{\underline{x}}^{(t)IN}$:

Let $\underline{a}^{na} = (\underline{a}_i)_{i=0,1,\dots,na-1}$ denote the parents of \underline{x} and $\underline{b}^{nb} = (\underline{b}_i)_{i=0,1,\dots,nb-1}$ its children.

Define

$$\pi_{\underline{x}}(x) = \sum_{a^{na}} P(x|a^{na}) \prod_i \pi_{\underline{x} \leftarrow \underline{a}_i}(a_i) \quad (35.55)$$

$$= E_{\underline{a}^{na}}[P(x|a^{na})] \quad (35.56)$$

(boundary case: if \underline{x} is a root node, use $\pi_{\underline{x}}(x) = P(x)$.) and

$$\lambda_{\underline{x}}(x) = \prod_i \lambda_{\underline{b}_i \Rightarrow \underline{x}}(x) . \quad (35.57)$$

(boundary case: if \underline{x} is a leaf node, use $\lambda_{\underline{x}}(x) = 1$.)

- RULE 1: (red parent)

From the $\lambda_{\underline{x} \Rightarrow \underline{a}}$ panel of Fig.35.11, we get

$$\underbrace{\lambda_{\underline{x} \Rightarrow \underline{a}_i}(a_i)}_{OUT} = \mathcal{N}(!a_i) \sum_x \left[\underbrace{\lambda_{\underline{x}}(x)}_{IN} \sum_{(a_k)_{k \neq i}} \left(P(x|a^{na}) \prod_{k \neq i} \underbrace{\pi_{\underline{x} \leftarrow \underline{a}_k}(a_k)}_{IN} \right) \right] \quad (35.58)$$

$$= \mathcal{N}(!a_i) \sum_x [\lambda_{\underline{x}}(x) E_{(a_k)_{k \neq i}}[P(x|a^{na})]] \quad (35.59)$$

$$= \mathcal{N}(!a_i) E_{(a_k)_{k \neq i}} E_{\underline{x}|a^{na}} \lambda_{\underline{x}}(x) \quad (35.60)$$

(boundary case: if \underline{x} is a root node, use $\lambda_{\underline{x} \Rightarrow \underline{a}_i}(a_i) = \mathcal{N}(!a_i)$.)

- **RULE 2: (red child)**

From the $\pi_{\underline{b} \leftarrow \underline{x}}$ panel of Fig.35.11, we get

$$\underbrace{\pi_{\underline{b}_i \leftarrow \underline{x}}(x)}_{OUT} = \mathcal{N}(!x) \underbrace{\pi_{\underline{x}}(x)}_{IN} \prod_{k \neq i} \underbrace{\lambda_{\underline{b}_k \Rightarrow \underline{x}}(x)}_{IN} \quad (35.61)$$

(boundary case: if \underline{x} is a leaf node, use $\pi_{\underline{b}_i \leftarrow \underline{x}}(x) = \mathcal{N}(!x)\pi_{\underline{x}}(x)$.)

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

$$BEL^{(t)}(x) = \mathcal{N}(!x)\lambda_{\underline{x}}^{(t)}(x)\pi_{\underline{x}}^{(t)}(x) . \quad (35.62)$$

Then

$$\lim_{t \rightarrow \infty} BEL^{(t)}(x) = P(x|\epsilon) . \quad (35.63)$$

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 $\underline{x} = x$.

35.4.1 How BP algo for polytrees reduces to the 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.(35.55) and (35.57) reduce to

$$\pi_{\underline{x}}(x) = \sum_a P(x|a)\pi_{\underline{x} \leftarrow \underline{a}}(a) \quad (35.64)$$

and

$$\lambda_{\underline{x}}(x) = \lambda_{\underline{b} \Rightarrow \underline{x}}(x) . \quad (35.65)$$

RULE 1 given by Eq.(35.58) reduces to

$$\lambda_{\underline{x} \Rightarrow \underline{a}}(a) = \mathcal{N}(!a) \sum_x \lambda_{\underline{x}}(x)P(x|a) \quad (35.66)$$

$$= \mathcal{N}(!a) \sum_x \lambda_{\underline{b} \Rightarrow \underline{x}}(x)P(x|a) \quad (35.67)$$

RULE 2 given by Eq.(35.61) reduces to

$$\pi_{\underline{b} \Leftarrow \underline{x}}(x) = \mathcal{N}(!x)\pi_{\underline{x}}(x) \quad (35.68)$$

$$= \sum_a P(x|a)\pi_{\underline{x} \Leftarrow \underline{a}}(a). \quad (35.69)$$

35.5 Derivation of BP Algorithm for Polytrees

This derivation is taken from the 1988 book Ref.[26] 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 $P(x|\epsilon)$.

$(\underline{a}_i)_{i=0,1,\dots,na-1}$. = parent nodes (mnemonic: a=ancestor) of \underline{x}

$(\underline{b}_i)_{i=0,1,\dots,nb-1}$. = 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.

$$\underline{\epsilon}_{\underline{x}}^- = \underline{\epsilon} \cap an(\underline{x}) \text{ (evidence in past of } \underline{x})^4$$

$$\epsilon_{\underline{x}a_i}^- = \epsilon_{\underline{x}}^- \cap an(\underline{a}_i).$$

$$\text{Note that } \epsilon_{\underline{x}}^- = \cup_i \epsilon_{\underline{x}a_i}^-$$

$$\underline{\epsilon}_{\underline{x}}^+ = \underline{\epsilon} \cap [de(\underline{x}) \cup \underline{x}] \text{ (evidence in future of } \underline{x})$$

$$\epsilon_{\underline{x}b_i}^+ = \epsilon_{\underline{x}}^+ \cap [de(\underline{b}_i) \cup \underline{b}_i].$$

$$\text{Note that } \epsilon_{\underline{x}}^+ = \cup_i \epsilon_{\underline{x}b_i}^+$$

$$\text{Note that } \underline{\epsilon} = \underline{\epsilon}_{\underline{x}}^+ \cup \underline{\epsilon}_{\underline{x}}^-$$

$$\pi_{\underline{x}}(x) = P(x|\epsilon_{\underline{x}}^-) \quad (35.70)$$

$$\pi_{\underline{x} \Leftarrow \underline{a}_i}(a_i) = P(a_i|\epsilon_{\underline{x}a_i}^-) \quad (35.71)$$

$$\pi_{\underline{b}_i \Leftarrow \underline{x}}(x) = P(x|\epsilon_{\underline{x}b_i}^-) \quad (35.72)$$

$$\lambda_{\underline{x}}(x) = P(\epsilon_{\underline{x}}^+|x) \quad (35.73)$$

$$\lambda_{\underline{b}_i \Rightarrow \underline{x}}(x) = P(\epsilon_{\underline{x}b_i}^+|x) \quad (35.74)$$

$$\lambda_{\underline{x} \Rightarrow \underline{a}_i}(a_i) = P(\epsilon_{\underline{x}a_i}^+|a_i) \quad (35.75)$$

Expansions of $\lambda_{\underline{x}}(x)$ and $\pi_{\underline{x}}(x)$ into products of single node messages.

⁴ Careful: Chapter 4 of Ref.[26] uses $-$ to indicate the future and $+$ to indicate the past. This is the opposite of our notation.

$$\underbrace{P(x|\epsilon_{\underline{x}}^-)}_{\pi_{\underline{x}}(x)} = P(x| \cup_i \epsilon_{\underline{x}a_i}^-) \quad (35.76)$$

$$= \sum_{a^{na}} P(x|a^{na}) P(a^{na}| \cup_i \epsilon_{\underline{x}a_i}^-) \quad (35.77)$$

$$= \sum_{a^{na}} P(x|a^{na}) \prod_i \underbrace{P(a_i|\epsilon_{\underline{x}a_i}^-)}_{\pi_{\underline{x} \leftarrow a_i}(a_i)} \quad (35.78)$$

$$\underbrace{P(\epsilon_{\underline{x}}^+|x)}_{\lambda_{\underline{x}}(x)} = \prod_i \underbrace{P(\epsilon_{\underline{x}b_i}^+|x)}_{\lambda_{\underline{b}_i \Rightarrow \underline{x}}(x)} \quad (35.79)$$

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} :

$$P(\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-|x) = P(\epsilon_{\underline{x}}^+|x)P(\epsilon_{\underline{x}}^-|x). \quad (35.80)$$

This observation is key to the proof of the following claim:

Claim 25

$$P(x|\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-) = P(\epsilon_{\underline{x}}^+|x)P(x|\epsilon_{\underline{x}}^-) \frac{1}{P(\epsilon_{\underline{x}}^+|\epsilon_{\underline{x}}^-)} \quad (35.81)$$

$$= \mathcal{N}(!x)P(\epsilon_{\underline{x}}^+|x)P(x|\epsilon_{\underline{x}}^-) \quad (35.82)$$

$$= \mathcal{N}(!x) (\epsilon_{\underline{x}}^+ \leftarrow x \leftarrow \epsilon_{\underline{x}}^-) \quad (35.83)$$

$$= \mathcal{N}(!x)\lambda_{\underline{x}}(x)\pi_{\underline{x}}(x) \quad (35.84)$$

proof:

$$P(x|\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-) = P(\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-|x) \frac{P(x)}{P(\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-)} \quad (35.85)$$

$$= P(\epsilon_{\underline{x}}^+|x)P(\epsilon_{\underline{x}}^-|x) \frac{P(x)}{P(\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-)} \quad (35.86)$$

$$= P(\epsilon_{\underline{x}}^+|x)P(x|\epsilon_{\underline{x}}^-) \frac{P(\epsilon_{\underline{x}}^-)}{P(\epsilon_{\underline{x}}^+, \epsilon_{\underline{x}}^-)} \quad (35.87)$$

$$= P(\epsilon_{\underline{x}}^+|x)P(x|\epsilon_{\underline{x}}^-) \frac{1}{P(\epsilon_{\underline{x}}^+|\epsilon_{\underline{x}}^-)} \quad (35.88)$$

QED

Next we prove BP rules 1 and 2.

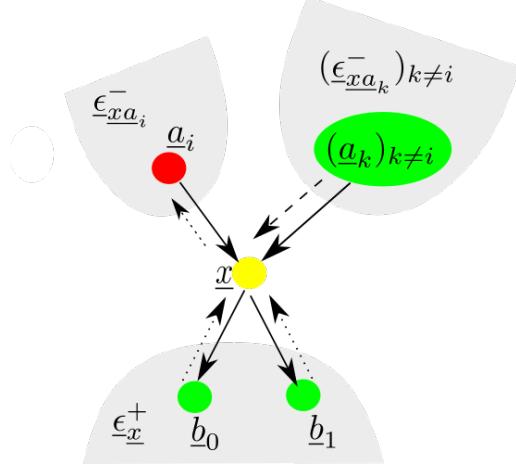


Figure 35.12: This figure is used in the derivation of the BP RULE 1.

- **RULE 1 (red parent)**

Note that

$$\epsilon_x^+ \cup \cup_{k \neq i} \epsilon_{xa_k}^- = (\epsilon_x^+ \cup \epsilon_x^-) - \epsilon_{xa_i}^- \quad (35.89)$$

$$= \epsilon_{xa_i}^+ \quad (35.90)$$

Let $y = (a_k)_{k \neq i}$ and $\epsilon_y^- = (\epsilon_{xa_k}^-)_{k \neq i}$.

$$\underbrace{P(\epsilon_{xa_i}^+ | a_i)}_{\lambda_{\underline{x} \Rightarrow \underline{a}_i}(a_i)} = P(\epsilon_x^+, \epsilon_y^- | a_i) \quad (35.91)$$

$$= \sum_x \sum_y P(\epsilon_x^+, \epsilon_y^- | x, y) P(x, y | a_i) \quad (35.92)$$

$$= \sum_x \sum_y P(\epsilon_x^+ | x) P(\epsilon_y^- | y) P(x | y, a_i) P(y | a_i) \quad (35.93)$$

$$= P(\epsilon_y^-) \sum_x \sum_y P(\epsilon_x^+ | x) \frac{P(y | \epsilon_y^-)}{P(y)} P(x | y, a_i) \underbrace{P(y | a_i)}_{= P(y)} \quad (35.94)$$

$$= \mathcal{N}(!a_i) \sum_x \sum_y P(\epsilon_x^+ | x) P(x | \underbrace{y, a_i}_{a^{na}}) P(y | \epsilon_y^-) \quad (35.95)$$

$$= \mathcal{N}(!a_i) \sum_x \underbrace{P(\epsilon_x^+ | x)}_{\lambda_{\underline{x}}(x)} \sum_{(a_k)_{k \neq i}} P(x | a^{na}) \prod_{k \neq i} \underbrace{P(a_k | \epsilon_{xa_k}^-)}_{\pi_{\underline{x} \Leftarrow \underline{a}_k}(a_k)} \quad (35.96)$$

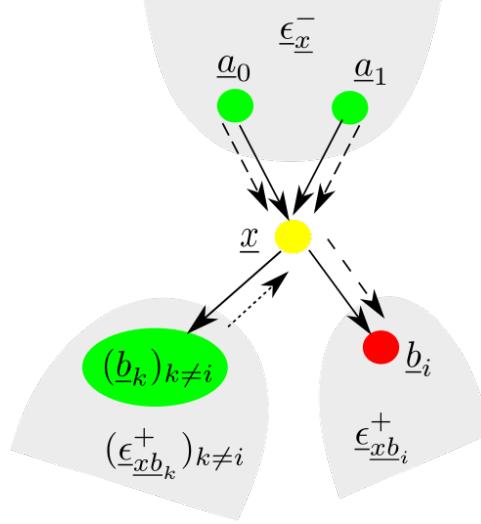


Figure 35.13: This figure is used in the derivation of the BP RULE 2.

- **RULE 2 (red child)**

Note that

$$(\cup_{k \neq i} \epsilon_{x b_k}^+) \cup \epsilon_x^- = (\epsilon_x^+ \cup \epsilon_x^-) - \epsilon_{x b_i}^+ \quad (35.97)$$

$$= \epsilon_{x b_i}^- \quad (35.98)$$

$$\underbrace{P(x | \epsilon_{x b_i}^-)}_{\pi_{b_i \leftarrow x}(x)} = P(x | (\epsilon_{x b_k}^+)_{k \neq i}, \epsilon_x^-) \quad (35.99)$$

$$= \mathcal{N}(!x) P((\epsilon_{x b_k}^+)_{k \neq i} | x) P(x | \epsilon_x^-) \quad (35.100)$$

$$= \mathcal{N}(!x) \left(\prod_{k \neq i} \underbrace{P(\epsilon_{x b_k}^+ | x)}_{\lambda_{b_k \Rightarrow x}(x)} \right) \underbrace{P(x | \epsilon_x^-)}_{\pi_x(x)} \quad (35.101)$$

35.6 Example of BP algo for a Tree

In this section, we describe how to apply the BP algo to the tree bnet Fig.35.14. In Fig.35.14, 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.35.14, the magenta 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.35.15. The steps encoded in the dynamic bnet are shown in Fig.35.16. Fig.35.16 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.

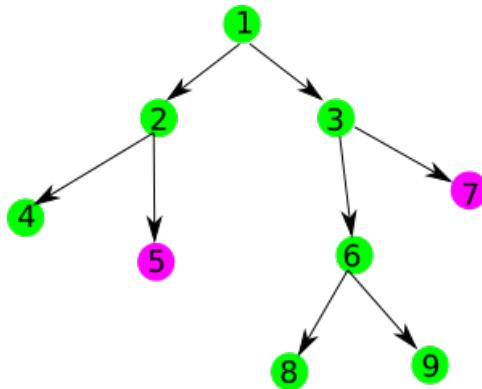


Figure 35.14: Example tree bnet used to illustrate BP.

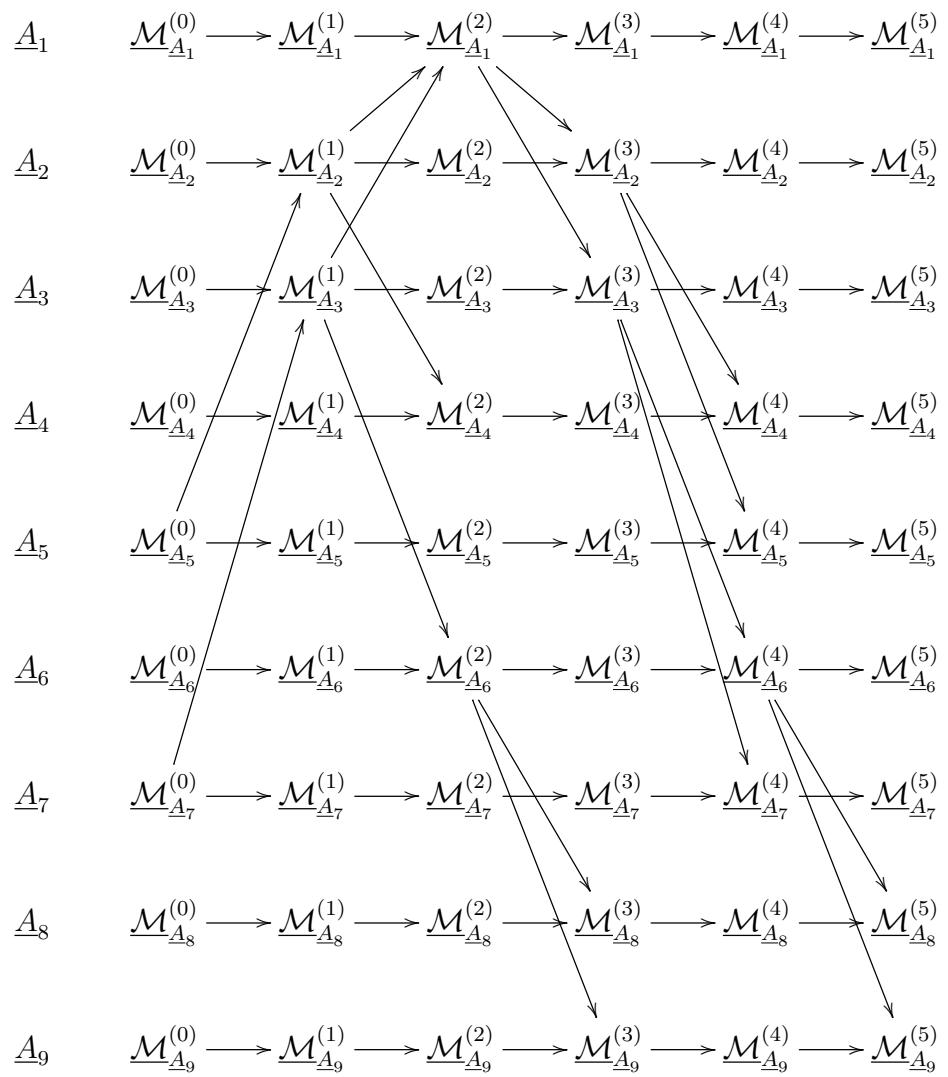


Figure 35.15: BP dynamic bnet for the bnet Fig.35.14.

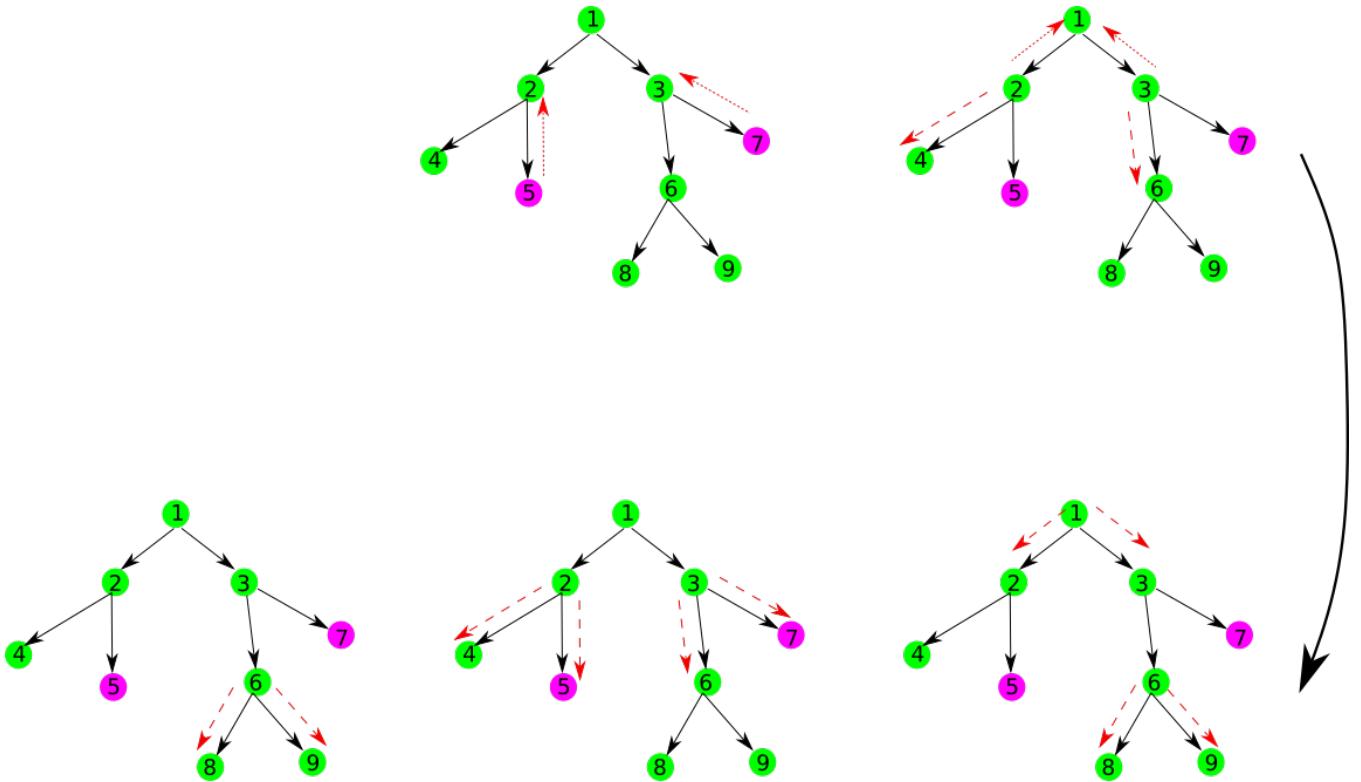


Figure 35.16: Steps encoded in the bnet Fig.35.15.

35.7 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 $f_{\alpha} : S_{\underline{x}_0} \times S_{\underline{x}_1} \times \dots \times S_{\underline{x}_{n_x-1}} \rightarrow \mathbb{R}$ for $\alpha = 0, 1, \dots, n_f - 1$. For instance, consider $g : S_{\underline{x}_0} \times S_{\underline{x}_1} \times S_{\underline{x}_2} \rightarrow \mathbb{R}$ defined by:

$$g(x_0, x_1, x_2) = f_0(x_0)f_1(x_0, x_1)f_2(x_0, x_1, x_2)f_3(x_1, x_2). \quad (35.102)$$

The **factor graph** for this function g is given by Fig.35.17.

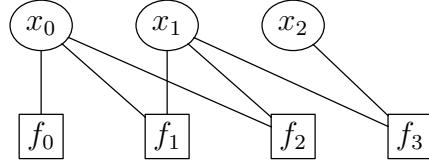


Figure 35.17: Factor graph for function g defined by Eq.(35.102).

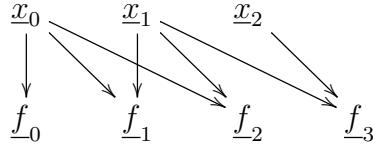


Figure 35.18: Bipartite bnet corresponding to factor graph Fig.35.17.

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 $i = 0, 1, \dots, n_x - 1$ and each f_{α} by \underline{f}_{α} for $\alpha = 0, 1, \dots, n_f - 1$. Then replace the connections (edges) of the factor graph by arrows from \underline{x}_i to \underline{f}_{α} . For example, Fig.35.18 is the image bipartite bnet of the source factor graph Fig.35.17.

Let $\underline{x}^{nx} = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{nx-1})$ and $\underline{f}^{nf} = (\underline{f}_0, \underline{f}_1, \dots, \underline{f}_{nf-1})$. Let⁵ $f_\alpha \in \{0, 1\}$ 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}_α in the image 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, as follows.

$$P(f_\alpha | \underline{x}^{nx}(\underline{f}_\alpha)) = y_\alpha \delta(f_\alpha, 1) + [1 - y_\alpha] \delta(f_\alpha, 0) \quad (35.103)$$

for $\alpha = 0, 1, \dots, nf - 1$ and

$$P_{\underline{x}_i}(x_i) = \text{arbitrary prior} \quad (35.104)$$

for $i = 0, 1, \dots, nx - 1$.

Note that

$$P(\underline{f}^{nf} = 1^{nf} | \underline{x}^{nx}) = \prod_\alpha f_\alpha(x_{nb(\underline{f}_\alpha)}) . \quad (35.105)$$

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

$$\underline{x} \longrightarrow \underline{y} \quad (35.106)$$

of the tree bnet by

$$\underline{x} \longrightarrow P_{\underline{y}|\underline{x}} \longleftarrow \underline{y} . \quad (35.107)$$

For example, the tree bnet Fig.35.19 has the image bipartite bnet given by Fig.35.20. The bnet Fig.35.21 is just a different way of drawing the bnet Fig.35.20.

The node TPMs, printed in blue, for the image bipartite bnet Fig.35.20, are as follows. We express the TPMs of the image bnet in terms of the TPMs of the source bnet Fig.35.19. Let

$$P(P_{\underline{y}|\underline{x}} | x, y) = P_{\underline{y}|\underline{x}}(y|x) \delta(P_{\underline{y}|\underline{x}}, 1) + (1 - P_{\underline{y}|\underline{x}}(y|x)) \delta(P_{\underline{y}|\underline{x}}, 0) \quad (35.108)$$

for all the leaf nodes $P_{\underline{y}|\underline{x}} \in \{0, 1\}$ of the image bipartite bnet. Also, let

$$P_{\underline{y}}(y) = \text{arbitrary prior} \quad (35.109)$$

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 be used to denote, besides a function and a boolean value, the real number $y_\alpha = f_\alpha(x_{nb(\underline{f}_\alpha)})$. However, we won’t be using it that third way in this chapter.

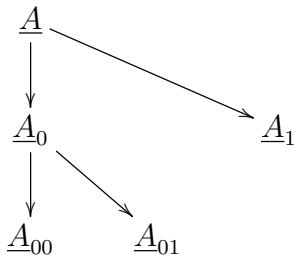


Figure 35.19: Example of a tree bnet.

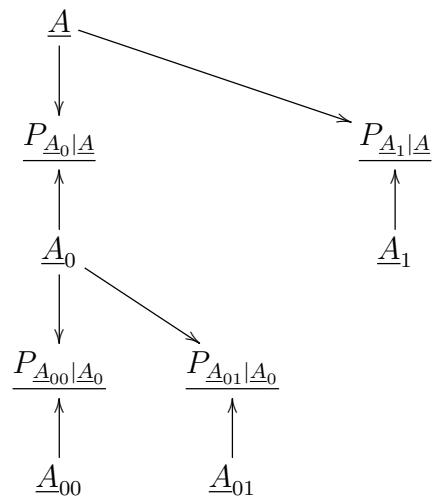


Figure 35.20: Bipartite bnet corresponding to tree bnet Fig.35.19.

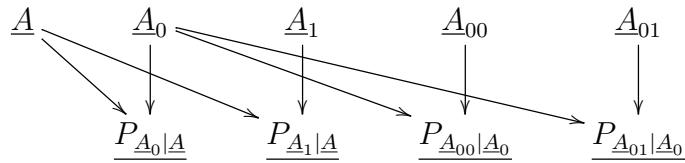


Figure 35.21: Different way of drawing the bnet Fig.35.20.

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,

$$P_{\underline{y}}(\underline{y}) = P_{\underline{A}}(\underline{y}) . \quad (35.110)$$

35.8 BP for bipartite bnets (BP-BB)

For a bipartite bnet as defined above, with root nodes \underline{x}_i and leaf nodes \underline{f}_α , let

$$nb(i) = \{\alpha : \underline{f}_\alpha \in nb(\underline{x}_i)\} , \quad (35.111)$$

$$nb(\alpha) = \{i : \underline{x}_i \in nb(\underline{f}_\alpha)\} , \quad (35.112)$$

$$m_{\alpha \leftarrow i}(x_i) = \pi_{\underline{f}_\alpha \leftarrow \underline{x}_i}(x_i) , \quad (35.113)$$

$$m_{\alpha \rightarrow i}(x_i) = \lambda_{\underline{f}_\alpha \rightarrow \underline{x}_i}(x_i) , \quad (35.114)$$

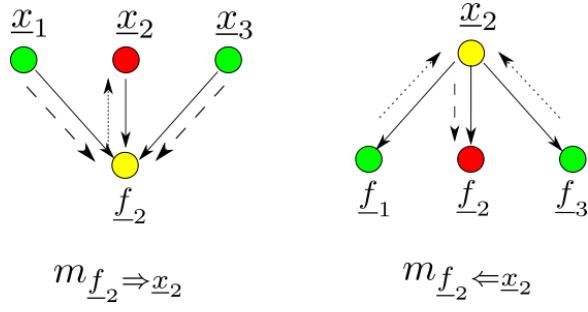


Figure 35.22: Fig.35.11 becomes this figure for the special case of a bipartite bnet. Union of green nodes and the red node = full neighborhood of yellow node. There are two possible cases: the red node is either a parent or a child of the yellow node.

Next we will show how to find $m_{\alpha \leftarrow i}^{(t)}$ and $m_{\alpha \rightarrow i}^{(t)}$ from $m_{\alpha \leftarrow i}^{(t-1)}$ and $m_{\alpha \rightarrow i}^{(t-1)}$.

1. Traversing an x (i.e., root) node.

See the $m_{\underline{f}_2 \leftarrow \underline{x}_2}$ panel of Fig.35.22.

For $i = 0, 1, \dots, nx - 1$, if $\alpha \in nb(i)$, then,

$$m_{\alpha \leftarrow i}^{(t)}(x_i) = \prod_{\beta \in nb(i) - \alpha} m_{\beta \rightarrow i}^{(t-1)}(x_i) , \quad (35.115)$$

whereas if $\alpha \notin nb(i)$

$$m_{\alpha \leftarrow i}^{(t)}(x_i) = m_{\alpha \leftarrow i}^{(t-1)}(x_i) . \quad (35.116)$$

2. Traversing an f (i.e., leaf) node.

See the $m_{\underline{f}_2 \Rightarrow \underline{x}_2}$ panel of Fig.35.22.

For $\alpha = 0, 1, \dots, nf - 1$, if $i \in nb(\alpha)$, then

$$m_{\alpha \Rightarrow i}^{(t)}(x_i) = \sum_{(x_k)_{k \in nb(\alpha)-i}} f_\alpha(x_{nb(\alpha)}) \prod_{k \in nb(\alpha)-i} m_{\alpha \Leftarrow k}^{(t-1)}(x_k) \quad (35.117)$$

$$= E_{(x_k)_{k \in nb(\alpha)-i}}^{(t-1)} [f_\alpha(x_{nb(\alpha)})] , \quad (35.118)$$

whereas if $i \notin nb(\alpha)$

$$m_{\alpha \Rightarrow i}^{(t)}(x_i) = m_{\alpha \Rightarrow i}^{(t-1)}(x_i) . \quad (35.119)$$

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:

$$P(x_i | \epsilon) = \lim_{t \rightarrow \infty} \mathcal{N}(!x_i) \prod_{\alpha \in nb(i)} m_{\alpha \Rightarrow i}^{(t)}(x_i) \quad (35.120)$$

and

$$P(x_{nb(\alpha)} | \epsilon) = \lim_{t \rightarrow \infty} \mathcal{N}(!x_{nb(\alpha)}) f_\alpha(x_{nb(\alpha)}) \prod_{k \in nb(\alpha)} m_{\alpha \Leftarrow k}^{(t)}(x_k) . \quad (35.121)$$

35.8.1 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.

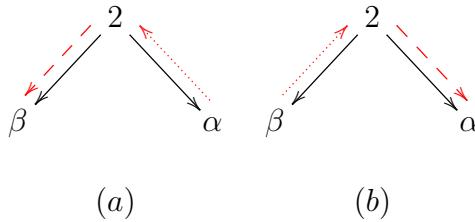


Figure 35.23: Traversing a root node of a Markov chain (a)Propagation towards left (i.e., towards future). (b)Propagation towards right (i.e., towards past).

Consider the BP-BB rule for traversing a root node. When traveling towards the left as in Fig.35.23 (a), it implies that

$$m_{\alpha \Rightarrow 2}(x_2) = m_{\beta \Leftarrow 2}(x_2) , \quad (35.122)$$

and when traveling towards the right as in Fig.35.23 (b), it implies that

$$m_{\beta \Rightarrow 2}(x_2) = m_{\alpha \Leftarrow 2}(x_2) . \quad (35.123)$$

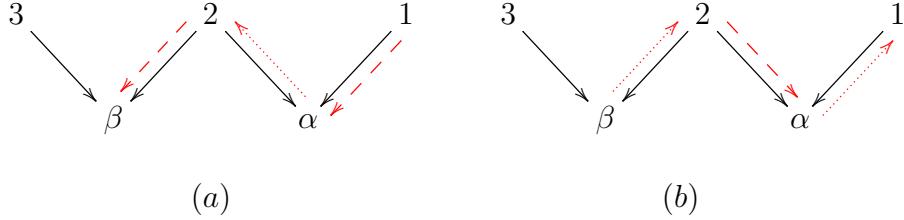


Figure 35.24: Traversing a leaf node of a Markov chain (a)Propagation towards left (i.e., towards future). (b)Propagation towards right (i.e., towards past).

Now consider the BP-BB rule for traversing a leaf node. When traveling to the left as in Fig.35.24 (a), it implies that

$$\underbrace{m_{\alpha \Rightarrow 2}(x_2)}_{\lambda} = \sum_{x_1} P(x_2|x_1) \underbrace{m_{\alpha \Leftarrow 1}(x_1)}_{\pi} . \quad (35.124)$$

One can rewrite the left and right hand sides (LHS, RHS) of Eq.(35.124) as follows

$$RHS = \sum_{x_1} P(x_2|x_1) \pi_{\alpha \Leftarrow 1}(x_1) , \quad (35.125)$$

and

$$LHS = m_{\alpha \Rightarrow 2}(x_2) = m_{\beta \Leftarrow 2}(x_2) = \pi_{\beta \Leftarrow 2}(x_2) , \quad (35.126)$$

Therefore

$$\pi_{\beta \Leftarrow 2}(x_2) \sum_{x_1} P(x_2|x_1) \pi_{\alpha \Leftarrow 1}(x_1) . \quad (35.127)$$

Once again, consider the BP-BB rule for traversing a leaf node. When traveling to the right as in Fig.35.24 (b), it implies that

$$\underbrace{m_{\alpha \Rightarrow 1}(x_1)}_{\lambda} = \sum_{x_2} P(x_2|x_1) \underbrace{m_{\alpha \Leftarrow 2}(x_2)}_{\pi} . \quad (35.128)$$

One can rewrite the left and right hand sides (LHS, RHS) of Eq.(35.128) as follows

$$RHS = \sum_{x_2} P(x_2|x_1) \pi_{\alpha \Leftarrow 2}(x_2) \quad (35.129)$$

$$= \sum_{x_2} P(x_2|x_1) \lambda_{\beta \Rightarrow 2}(x_2) , \quad (35.130)$$

and

$$LHS = \lambda_{\alpha \Rightarrow 1}(x_1) . \quad (35.131)$$

Therefore,

$$\lambda_{\alpha \Rightarrow 1}(x_1) = \sum_{x_2} P(x_2|x_1) \lambda_{\beta \Rightarrow 2}(x_2) . \quad (35.132)$$

Finally, note that Eq.(35.120) becomes

$$P(x_2|\epsilon) = \mathcal{N}(!x_2) m_{\beta \Rightarrow 2}(x_2) m_{\alpha \Rightarrow 2}(x_2) \quad (35.133)$$

$$= \mathcal{N}(!x_2) m_{\alpha \Leftarrow 2}(x_2) m_{\alpha \Rightarrow 2}(x_2) \quad (35.134)$$

$$= \mathcal{N}(!x_2) \pi_{\alpha \Leftarrow 2}(x_2) \lambda_{\alpha \Rightarrow 2}(x_2) \quad (35.135)$$

$$= \mathcal{N}(!x_2) P(x_2|\epsilon^-) P(x_2|\epsilon^+) \quad (35.136)$$

and Eq.(35.121) becomes

$$P(x_2, x_1) = \mathcal{N}(!x_2, !x_1) P(x_2|x_1) m_{\alpha \Leftarrow 1}(x_1) m_{\alpha \Leftarrow 2}(x_2) \quad (35.137)$$

$$= \mathcal{N}(!x_2, !x_1) P(x_2|x_1) \pi_{\alpha \Leftarrow 1}(x_1) \pi_{\alpha \Leftarrow 2}(x_2) . \quad (35.138)$$

35.8.2 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.

Applying to the left panel of Fig.35.25 the BP-BB rule for traversing a root node, we get

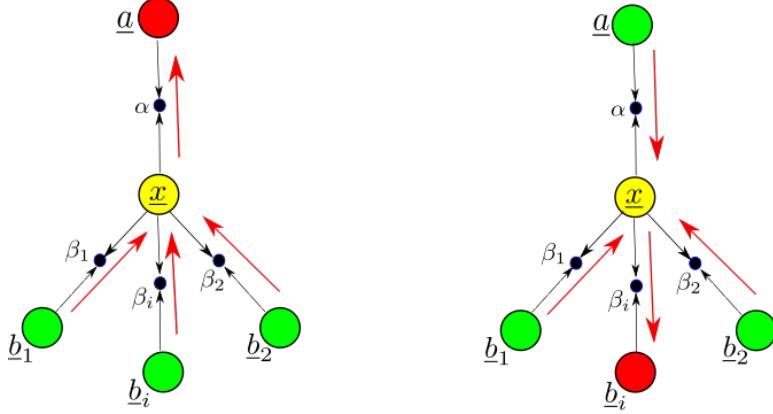
$$m_{\alpha \Leftarrow \underline{x}}(x) = \prod_i m_{\beta_i \Rightarrow \underline{x}}(x) . \quad (35.139)$$

Applying to the left panel of Fig.35.25 the BP-BB rule for traversing a leaf node, we get

$$m_{\alpha \Rightarrow \underline{a}}(a) = \mathcal{N}(!a) \sum_x m_{\alpha \Leftarrow \underline{x}}(x) P(x|a) . \quad (35.140)$$

Combining Eqs.(35.139) and (35.140), we get

$$m_{\alpha \Rightarrow \underline{a}}(a) = \mathcal{N}(!a) \sum_x P(x|a) \prod_i m_{\beta_i \Rightarrow \underline{x}}(x) , \quad (35.141)$$



$$\lambda_{\underline{x} \Rightarrow \underline{a}} \quad \pi_{\underline{b} \Leftarrow \underline{x}}$$

Figure 35.25: Subgraph of a tree bnet. This is the same as Fig.35.11, 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.35.11. 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.

which can be rewritten as

$$\lambda_{\underline{x} \Rightarrow \underline{a}}(a) = \mathcal{N}(!a) \sum_x P(x|a) \underbrace{\prod_i \lambda_{b_i \Rightarrow \underline{x}}(x)}_{\lambda_{\underline{x}}(x)} . \quad (35.142)$$

Eq.35.142 is just RULE 1 for general BP.

Applying to the right panel of Fig.35.25 the BP-BB rule for traversing a root node, we get

$$m_{\beta_i \Leftarrow \underline{x}}(x) = \mathcal{N}(!x)m_{\alpha \Rightarrow \underline{x}}(x) \prod_{k \neq i} m_{\beta_k \Rightarrow \underline{x}}(x) \quad (35.143)$$

Applying to the right panel of Fig.35.25 the BP-BB rule for traversing a leaf node, we get

$$m_{\alpha \Rightarrow \underline{x}}(x) = \sum_a P(x|a)m_{\alpha \Leftarrow a}(a) \quad (35.144)$$

$$= \sum_a P(x|a)\pi_{\underline{x} \Leftarrow a}(a) \quad (35.145)$$

$$= \pi_{\underline{x}}(x) . \quad (35.146)$$

Combining Eqs.(35.143) and (35.146), we get

$$\pi_{\underline{b}_i \leftarrow \underline{x}}(x) = \mathcal{N}(!x)\pi_{\underline{x}}(x) \prod_{k \neq i} \lambda_{\underline{b}_k \Rightarrow \underline{x}}(x) . \quad (35.147)$$

Eq.(35.147) is just RULE 2 of general BP.

35.9 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 nature⁶ of the decomposition. I prefer to call it a **recursive sum of products (RSOP) decomposition**, and will call it so henceforth in this chapter.

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 $\underline{x}_{n-1} \leftarrow \dots \leftarrow \underline{x}_1 \leftarrow \underline{x}_0$, where $x_i \in \{0, 1, 2\}$ for all i . Note that if we calculate $P(x_{n-1})$ as follows

$$P(x_{n-1}) = \left[\sum_{x_{n-2}} P(x_{n-1}|x_{n-2}) \dots \left[\sum_{x_1} P(x_2|x_1) \left[\sum_{x_0} P(x_1|x_0)P(x_0) \right] \dots \right] \right] , \quad (35.148)$$

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

$$P(x_{n-1}) = \sum_{x_{n-2}} \dots \sum_{x_1} \sum_{x_0} P(x_{n-1}|x_{n-2}) \dots P(x_2|x_1)P(x_1|x_0)P(x_0) , \quad (35.149)$$

we need to perform $3^n - 1$ additions and $3^n(n - 1)$ multiplications.

⁶ By "recursive nature", we mean bootstrapped 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.

Chapter 36

Missing Data, Imputation

This chapter assumes that the reader has read some parts of Chapter 18 on the Expectation Maximization (EM) algo and Chapter 33 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	
		0		1	
		1		0	(1,0,1)
		1		1	
5	NA	0	0	1	(0,1,0)
6	NA	0	0	1	(0,0,0)

Table 36.1: **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** $\vec{x} = (x[\sigma])_{\sigma=0,1,\dots,nsam-1}$ where $x = (x_0, x_1, \dots, x_{nx-1})$ 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: $\vec{a} = (\underline{a}[\sigma])_{\sigma=0,1,\dots,nsam-1}$, 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 36.1.

36.1 Imputation via EM

We begin by augmenting Fig.18.1 (the first figure in Chapter 18) by adding to it a new node \vec{m} called the **missingness variable**. Recall that node θ represents the **unknown parameters**, node \vec{x} represents the **observed variables**, and node \vec{h} represents the **latent variables**. Both θ and \vec{h} are hidden (i.e., unobserved). Fig.36.1 shows 3 popular ways of connecting node \vec{m} to the other nodes in the graph Fig.18.1.

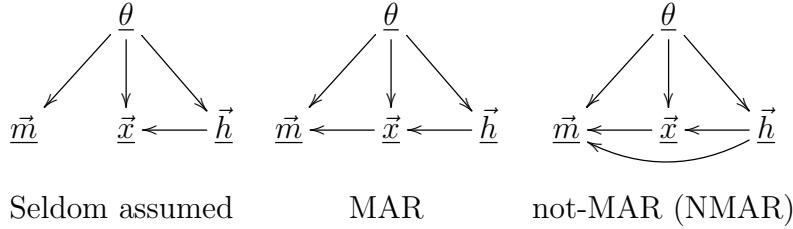


Figure 36.1: The left bnet is seldom assumed. The middle bnet is referred to as the MAR (missing at random) assumption. The right bnet is referred to as the not-MAR (NMAR) assumption.

From Fig.36.1, we have

$$P(\vec{m}|\vec{x}, \vec{h}, \theta) = \begin{cases} P(\vec{m}|\theta) & \text{Seldom assumed. Called missing-CAR (MCAR)} \\ P(\vec{m}|\vec{x}, \theta) & \text{MAR} \\ P(\vec{m}|\vec{x}, \vec{h}, \theta) & \text{not-MAR (NMAR)} \end{cases} \quad (36.1)$$

For doing imputation via EM, we connect node \vec{m} as shown in the middle bnet (called MAR) of Fig.36.1.

For the example of Table 36.1, we have variables \vec{m}, \vec{x} and \vec{h} whose values range over the following sets:

$$\begin{aligned} \vec{x} &= (\underline{x}_0, \underline{x}_1, \underline{x}_2) \\ \vec{h} &= (\underline{h}_0) \\ \underline{h}_0[\sigma] &\in \{0, 1\}, \\ \underline{x}_i[\sigma] &\in \{0, 1\} \text{ for } i = 0, 1, 2, \end{aligned}$$

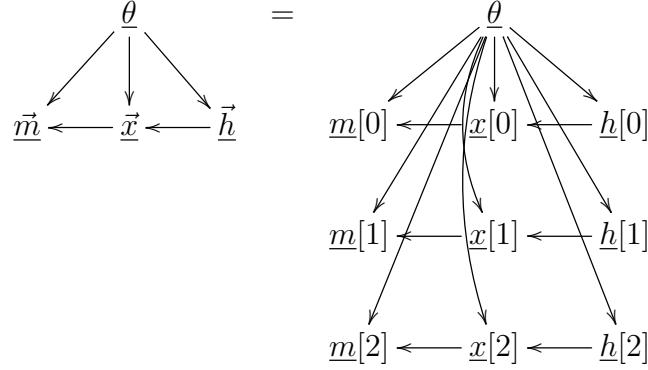


Figure 36.2: MAR bnet with $nsam = 3$.

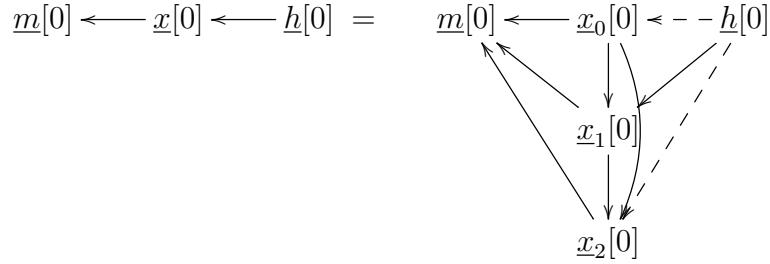


Figure 36.3: Our example for imputation via EM assumes this bnet between nodes $m[\sigma], \underline{x}[\sigma], \underline{h}[\sigma]$.

$$\underline{m}_i[\sigma] \in \{0, 1\} \text{ for } i = 0, 1, 2.$$

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.36.3. 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.36.3 without the dashed arrows, one can include those arrows too, at the expense of more unknown parameters (i.e., degrees of freedom) to be lumped into θ . We will parameterize the TPMs corresponding to Fig.36.3 using a Categorical Distribution for each column of the TPMs. We will thus assume that the bnet of Fig.36.3 has the following TPMs, printed in blue.

$$P(h_0^{[\sigma]} | \theta) = \begin{array}{c|cc} & 1 - \theta_0 & \\ \hline 1 & \theta_0 & \end{array} \quad (36.2)$$

$$P(x_0^{[\sigma]} | \theta) = \begin{array}{c|cc} & 1 - \theta_1 & \\ \hline 0 & \theta_1 & \end{array} \quad (36.3)$$

$$P(x_1^{[\sigma]} | x_0^{[\sigma]}, h^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1 - \theta_2 & 1 - \theta_3 & 1 - \theta_4 & 1 - \theta_5 \\ 1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{array} \quad (36.4)$$

$$P(x_2^{[\sigma]} | x_1^{[\sigma]}, x_0^{[\sigma]}, h^{[\sigma]}, \theta) = \begin{array}{c|cccc} & 00 & 01 & 10 & 11 \\ \hline 0 & 1 - \theta_6 & 1 - \theta_7 & 1 - \theta_8 & 1 - \theta_9 \\ 1 & \theta_6 & \theta_7 & \theta_8 & \theta_9 \end{array} \quad (36.5)$$

$$P(m^{[\sigma]} | x^{[\sigma]}, \theta) = \frac{1}{nsam} P((x_i)_{\forall i \ni m_i=1} | (x_i)_{\forall i \ni m_i=0}, \theta) \quad (36.6)$$

Eq.(36.6) can be illustrated as follows. In Table 36.2, we added a $P(m)$ column to Table 36.1.

	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		$\frac{1}{nsam} P(x_0 = 0, x_2 = 0 x_1 = 1, \theta)$
		0		1		$\frac{1}{nsam} P(x_0 = 0, x_2 = 1 x_1 = 1, \theta)$
		1		0		$\frac{1}{nsam} P(x_0 = 1, x_2 = 0 x_1 = 1, \theta)$
		1		1		$\frac{1}{nsam} P(x_0 = 1, x_2 = 1 x_1 = 1, \theta)$
5	NA	0	0		(1,0,1)	$\frac{1}{nsam} P(x_1 = 0 x_0 = 0, x_2 = 1, \theta)$
		1		1		$\frac{1}{nsam} P(x_1 = 1 x_0 = 0, x_2 = 1, \theta)$
6	NA	0	0	1	(0,0,0)	$\frac{1}{nsam}$

Table 36.2: $P(m)$ column added to Table 36.1. Note that $\sum_m P(m) = 1$.

$$\theta = (\theta_i)_{i=0,1,\dots,9} \quad (36.7)$$

$$P(m^{[\sigma]}, x^{[\sigma]}, h^{[\sigma]} | \theta) = P(m^{[\sigma]} | x^{[\sigma]}, \theta) P(x^{[\sigma]} | h^{[\sigma]}, \theta) P(h^{[\sigma]} | \theta) \quad (36.8)$$

$$P(x^{[\sigma]} | h^{[\sigma]}, \theta) = P(x_2^{[\sigma]} | x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) P(x_1^{[\sigma]} | x_0^{[\sigma]}, h^{[\sigma]}, \theta) P(x_0^{[\sigma]} | \theta) \quad (36.9)$$

$$P(x_1^{[\sigma]} | x_0^{[\sigma]}, \theta) = \sum_h P(x_1^{[\sigma]} | x_0^{[\sigma]}, h^{[\sigma]}, \theta) P(h^{[\sigma]} | \theta) \quad (36.10)$$

$$P(x^{[\sigma]} | \theta) = P(x_2^{[\sigma]} | x_1^{[\sigma]}, x_0^{[\sigma]}, \theta) P(x_1^{[\sigma]} | x_0^{[\sigma]}, \theta) P(x_0^{[\sigma]} | \theta) \quad (36.11)$$

$$Q(\theta|\theta^{(t)}) = \sum_{\vec{m}, \vec{h}} P(\vec{m}, \vec{h} | \vec{x}, \theta^{(t)}) \ln P(\vec{m}, \vec{x}, \vec{h} | \theta) \quad (36.12)$$

$$= \sum_{\vec{m}, \vec{h}} \left[\prod_{\sigma} P(m^{[\sigma]}, h^{[\sigma]} | x^{[\sigma]}, \theta^{(t)}) \right] \ln \left[\prod_{\sigma} P(m^{[\sigma]}, x^{[\sigma]}, h^{[\sigma]} | \theta) \right] \quad (36.13)$$

$$= \sum_{\sigma} \sum_{m^{[\sigma]}, h^{[\sigma]}} P(m^{[\sigma]}, h^{[\sigma]} | x^{[\sigma]}, \theta^{(t)}) \ln P(m^{[\sigma]}, x^{[\sigma]}, h^{[\sigma]} | \theta) \quad (36.14)$$

$$= \sum_{\sigma} \sum_{m^{[\sigma]}, h^{[\sigma]}} \frac{P(m^{[\sigma]}, h^{[\sigma]}, x^{[\sigma]} | \theta^{(t)})}{P(x^{[\sigma]} | \theta^{(t)})} \ln P(m^{[\sigma]}, x^{[\sigma]}, h^{[\sigma]} | \theta) \quad (36.15)$$

Once you find optimal parameters θ^* by recursing this $Q(\theta|\theta^{(t)})$, you can evaluate numerically the $P(m)$ column of Table 36.2. In Table 36.2, 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.

36.2 Imputation via MCMC

A simple and popular way to do imputation via MCMC is described in Ref.[38]. It goes as follows.

Let

$$\underline{H}^{[\sigma]} = (\underline{h}^{[\sigma]}, \underline{m}^{[\sigma]}) \quad (36.16)$$

for $\sigma = 0, 1, \dots, nsam - 1$. Initialize $\theta^{(0)}$ to a random value within the allowed ranges. Do the following 2 steps, for $t = 0, 1, \dots, T - 1$, 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 $\sigma = 0, 1, \dots, nsam - 1$, find a sample

$$(H^{[\sigma]})^{(t+1)} \sim P(H^{[\sigma]} | x^{[\sigma]}, \theta^{(t)}) . \quad (36.17a)$$

- **STEP 2:** Find a sample

$$\theta^{(t+1)} \sim P^{(t+1)}(\theta) \quad (36.17b)$$

where

$$P^{(t+1)}(\theta) = \mathcal{N}(!\theta) P(\vec{x}, \vec{H}^{(t+1)} | \theta) \quad (36.17c)$$

$$= \mathcal{N}(!\theta) \prod_{\sigma} P(x^{[\sigma]}, (H^{[\sigma]})^{(t+1)} | \theta) . \quad (36.17d)$$

Fig.36.4 illustrates this two step recursive process using a bnet.

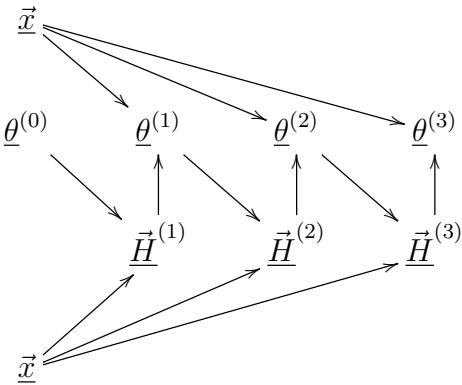


Figure 36.4: bnet illustrating Eqs.(36.17) for doing imputation via MCMC. The **same** node $\underline{\vec{x}}$ appears twice to make the graph clearer.

36.3 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 37

Monty Hall Problem

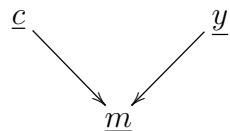


Figure 37.1: Monty Hall Problem.

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.37.1, 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 $S_c = S_y = S_m = \{1, 2, 3\}$.

Node matrices printed in blue:

$$P(c) = \frac{1}{3} \text{ for all } c \quad (37.1)$$

$$P(y) = \frac{1}{3} \text{ for all } y \quad (37.2)$$

$$P(m|c, y) = \mathbb{1}(m \neq c) \left[\frac{1}{2} \mathbb{1}(y = c) + \mathbb{1}(y \neq c) \mathbb{1}(m \neq y) \right] \quad (37.3)$$

It's easy to show that the above node probabilities imply that

$$P(c = 1|m = 2, y = 1) = \frac{1}{3} \quad (37.4)$$

$$P(c = 3|m = 2, y = 1) = \frac{2}{3} \quad (37.5)$$

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 38

Naive Bayes

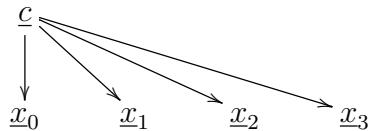


Figure 38.1: bnet for Naive Bayes with 4 features

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 $i = 0, 1, 2, \dots, F - 1$. F =number of features.

Define

$$x. = [x_0, x_1, \dots, x_{F-1}] . \quad (38.1)$$

For the bnet of Fig.38.1,

$$P(c, x.) = P(c) \prod_{i=0}^{F-1} P(x_i|c) . \quad (38.2)$$

Given $x.$ values, find most likely class $c \in S_{\underline{c}}$.

Maximum a Posteriori (MAP) estimate:

$$c^* = \operatorname{argmax}_c P(c|x.) \quad (38.3)$$

$$= \operatorname{argmax}_c \frac{P(c, x.)}{P(x.)} \quad (38.4)$$

$$= \operatorname{argmax}_c P(c, x.) . \quad (38.5)$$

Chapter 39

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 .

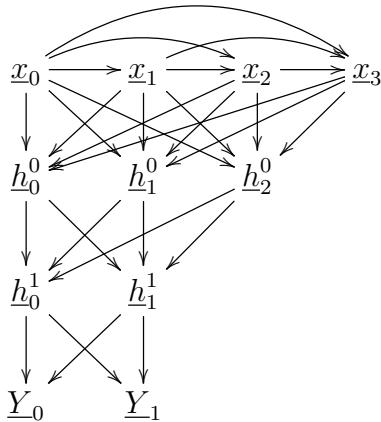


Figure 39.1: Neural Network (feed forward) with 4 layers: input layer $\underline{x}.$, 2 hidden layers $\underline{h}^0.$, $\underline{h}^1.$ and output layer $\underline{Y}.$

Consider Fig.39.1.

$\underline{x}_i \in \{0, 1\}$ for $i = 0, 1, 2, \dots, nx - 1$ is the **input layer**.

$\underline{h}_i^\lambda \in \mathbb{R}$ for $i = 0, 1, 2, \dots, nh(\lambda) - 1$ is the λ -th **hidden layer**. $\lambda = 0, 1, 2, \dots, \Lambda - 2$. 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 $i = 0, 1, 2, \dots, ny - 1$ 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 $i = 0, 1, \dots, ny - 1$. $Y = \hat{y}$ 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.39.1 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 $w_{i|j}^\lambda, b_i^\lambda \in \mathbb{R}$ 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.39.1:

$$P(x_i | x_{i-1}, x_{i-2}, \dots, x_0) = \text{given} \quad (39.1)$$

$$P(h_i^\lambda | h_{\cdot}^{\lambda-1}) = \delta \left(h_i^\lambda, \mathcal{A}_i^\lambda \left(\sum_j w_{i|j}^\lambda h_j^{\lambda-1} + b_i^\lambda \right) \right), \quad (39.2)$$

where $P(h_i^0 | h^{-1}) = P(h_i^0 | x)$.

$$P(Y_i | h_{\cdot}^{\Lambda-2}) = \delta \left(Y_i, \mathcal{A}_i^{\Lambda-1} \left(\sum_j w_{i|j}^{\Lambda-1} h_j^{\Lambda-2} + b_i^{\Lambda-1} \right) \right). \quad (39.3)$$

39.1 Activation Functions $\mathcal{A}_i^\lambda : \mathbb{R} \rightarrow \mathbb{R}$

Activation functions must be nonlinear.

- Step function (Perceptron)

$$\mathcal{A}(x) = \mathbb{1}(x > 0) \quad (39.4)$$

Zero for $x \leq 0$, one for $x > 0$.

- Sigmoid function

$$\mathcal{A}(x) = \frac{1}{1 + e^{-x}} = \text{smoid}(x) \quad (39.5)$$

Smooth, monotonically increasing function. $\text{smoid}(-\infty) = 0, \text{smoid}(0) = 0.5, \text{smoid}(\infty) = 1$.

$$\text{smoid}(x) + \text{smoid}(-x) = \frac{1}{1+e^{-x}} + \frac{1}{1+e^x} \quad (39.6)$$

$$= \frac{2+e^x+e^{-x}}{2+e^x+e^{-x}} \quad (39.7)$$

$$= 1 \quad (39.8)$$

- **Hyperbolic tangent**

$$\mathcal{A}(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (39.9)$$

Smooth, monotonically increasing function. $\tanh(-\infty) = -1$, $\tanh(0) = 0$, $\tanh(\infty) = 1$.

Odd function:

$$\tanh(-x) = -\tanh(x) \quad (39.10)$$

Whereas $\text{smoid}(x) \in [0, 1]$, $\tanh(x) \in [-1, 1]$.

- **ReLU (Rectified Linear Unit)**

$$\mathcal{A}(x) = x \mathbb{1}(x > 0) = \max(0, x) . \quad (39.11)$$

Compare this to the step function.

- **Swish**

$$\mathcal{A}(x) = x \text{smoid}(x) \quad (39.12)$$

- **Softmax**

$$\mathcal{A}(x_i|x.) = \frac{e^{x_i}}{\sum_i e^{x_i}} \quad (39.13)$$

It's called softmax because if we approximate the exponentials, both in the numerator and denominator of Eq.(39.13), by the largest one, we get

$$\mathcal{A}(x_i|x.) \approx \mathbb{1}(x_i = \max_k x_k) . \quad (39.14)$$

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 ,

$$\mathcal{A}(x_0|x.) = \frac{e^{x_0}}{e^{x_0} + e^{x_1}} \quad (39.15)$$

$$= \text{smoid}(x_0 - x_1) , \quad (39.16)$$

$$\mathcal{A}(x_1|x.) = \text{smoid}(x_1 - x_0) . \quad (39.17)$$

39.2 Weight optimization via supervised training and gradient descent

The bnet of Fig.39.1 is used for classification of a single data point x . It assumes that the weights $w_{ij}^\lambda, b_i^\lambda$ are given.

To find the optimum weights via supervised training and gradient descent, one uses the bnet Fig.39.2.

In Fig.39.2, the nodes in Fig.39.1 become sampling space vectors. For example, \underline{x} becomes \vec{x} , where the components of \vec{x} in sampling space are $\underline{x}[\sigma] \in \{0, 1\}^{nx}$ for $\sigma = 0, 1, \dots, nsam(\vec{x}) - 1$.

$nsam(\vec{x})$ is the number of samples used to calculate the gradient during each **stage (aka iteration)** of Fig.39.2. We will also refer to $nsam(\vec{x})$ as the **mini-batch size**. A **mini-batch** is a subset of the training dataset.

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

$$W_{ij}^\lambda = [w_{ij}^\lambda, b_i^\lambda] . \quad (39.18)$$

These are the TPMs, printed in blue, for the nodes of the bnet Fig.39.2:

$$P(x.[\sigma]) = \text{given} . \quad (39.19)$$

$$P(y.[\sigma] | x.[\sigma]) = \text{given} . \quad (39.20)$$

$$P(h_i^\lambda[\sigma] | h_j^{\lambda-1}[\sigma]) = \delta \left(h_i^\lambda[\sigma], \mathcal{A}_i^\lambda \left(\sum_j w_{ij}^\lambda h_j^{\lambda-1}[\sigma] + b_i^\lambda \right) \right) \quad (39.21)$$

$$P(Y_i[\sigma] | h_j^{\Lambda-2}[\sigma]) = \delta \left(Y_i[\sigma], \mathcal{A}_i^{\Lambda-1} \left(\sum_j w_{ij}^{\Lambda-1} h_j^{\Lambda-2}[\sigma] + b_i^{\Lambda-1} \right) \right) \quad (39.22)$$

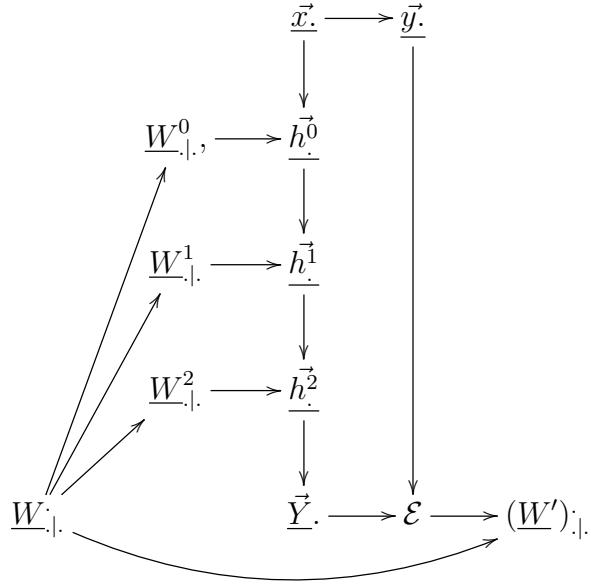


Figure 39.2: bnet for finding optimum weights of the bnet Fig.39.1 via supervised training and gradient descent.

$$P(W_{\cdot| \cdot}) = \text{given} \quad (39.23)$$

The first time it is used, $W_{\cdot| \cdot}$ is arbitrary. After the first time, it is determined by previous stage.

$$P(W_{\cdot| \cdot}^\lambda | W_{\cdot| \cdot}) = \delta(W_{\cdot| \cdot}^\lambda, (W_{\cdot| \cdot}))^\lambda \quad (39.24)$$

$$P(\mathcal{E} | \vec{y}., \vec{Y}.) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} \sum_i d(y_i[\sigma], Y_i[\sigma]) , \quad (39.25)$$

where

$$d(y, Y) = |y - Y|^2 . \quad (39.26)$$

If $y, Y \in [0, 1]$, one can use this instead

$$d(y, Y) = XE(y \rightarrow Y) = -y \ln Y - (1 - y) \ln(1 - Y) . \quad (39.27)$$

$$P((W')_{i|j}^\lambda | \mathcal{E}, W_{\cdot| \cdot}) = \delta((W')_{i|j}^\lambda, W_{i|j}^\lambda - \eta \partial_{W_{i|j}^\lambda} \mathcal{E}) \quad (39.28)$$

$\eta > 0$ 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 $\Delta \mathcal{E} = \frac{-1}{\eta} (\Delta W)^2 < 0$.

39.3 Non-dense layers

The TPM for a non-dense layer is of the form:

$$P(h_i^\lambda[\sigma] | h_{\cdot}^{\lambda-1}[\sigma]) = \delta(h_i^\lambda[\sigma], H_i^\lambda[\sigma]), \quad (39.29)$$

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.[36]. To dropout nodes from a fixed layer λ : For all i of layer λ , define a new node \underline{r}_i^λ with an arrow $\underline{r}_i^\lambda \rightarrow \underline{h}_i^\lambda$. For $r \in \{0, 1\}$, and some $p \in (0, 1)$, define

$$P(r_i^\lambda = r) = [p]^r [1 - p]^{1-r} \text{ (Bernouilli dist.)}. \quad (39.30)$$

Now one has

$$P(h_i^\lambda[\sigma] | h_{\cdot}^{\lambda-1}[\sigma], r_i^\lambda) = \delta(h_i^\lambda[\sigma], H_i^\lambda[\sigma]), \quad (39.31)$$

where

$$H_i^\lambda[\sigma] = \mathcal{A}_i^\lambda(r_i^\lambda \sum_j w_{i|j}^\lambda h_j^{\lambda-1}[\sigma] + b_i^\lambda). \quad (39.32)$$

This reduces overfitting. 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^λ nodes that induce dropout are only used in the training bnet Fig.39.2, not in the classification bnet Fig.39.1. We prefer to remove the \underline{r}_i^λ stochasticity from classification and for Fig.39.1 to act as an average over sampling space of Fig.39.2. Therefore, if weights $w_{i|j}^\lambda$ are obtained for a dropout layer λ in Fig.39.2, then that layer is used in Fig.39.1 with no \underline{r}_i^λ nodes but with weights $\langle r_i^\lambda \rangle w_{i|j}^\lambda = pw_{i|j}^\lambda$.

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 $\mathcal{F} : \{0, 1, \dots, nf - 1\} \rightarrow \mathbb{R}$.

σ =stride length

For $i \in \{0, 1, \dots, nh(\lambda) - 1\}$, let

$$H_i^\lambda[\sigma] = \sum_{j=0}^{nf-1} h_{j+i\sigma}^{\lambda-1}[\sigma] \mathcal{F}(j) . \quad (39.33)$$

For the indices not to go out of bounds in Eq.(39.33), we must have

$$nh(\lambda - 1) - 1 = nf - 1 + (nh(\lambda) - 1)\sigma \quad (39.34)$$

so

$$nh(\lambda) = \frac{1}{\sigma}[nh(\lambda - 1) - nf] + 1 . \quad (39.35)$$

- 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 $\{0, 1, \dots, nh(\lambda - 1) - 1\}$ into $nh(\lambda)$ mutually disjoint, nonempty sets called $Pool(i)$, where $i \in \{0, 1, \dots, nh(\lambda) - 1\}$.

- AvgPool

$$H_i^\lambda[\sigma] = \frac{1}{|Pool(i)|} \sum_{j \in Pool(i)} h_j^{\lambda-1}[\sigma] \quad (39.36)$$

- MaxPool

$$H_i^\lambda[\sigma] = \max_{j \in Pool(i)} h_j^{\lambda-1}[\sigma] \quad (39.37)$$

39.4 Autoencoder NN

If the sequence

$$nx, nh(0), nh(1), \dots, nh(\Lambda - 2), ny \quad (39.38)$$

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 40

Noisy-OR gate

The Noisy-OR gate was first proposed by Judea Pearl in his 1988 book Ref.[26].

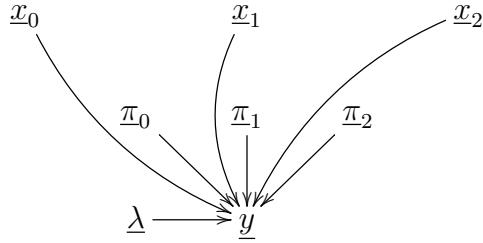


Figure 40.1: Noisy-OR gate $\underline{y} \in \{0, 1\}$ with $n = 3$, Boolean inputs $(\underline{x}_i)_{i=0,1,2}$ and parameters $\underline{\lambda}, (\underline{\pi})_{i=0,1,2}$.

Let

$\underline{\lambda} \in [0, 1]$ =gate leakage.

$y \in \{0, 1\}$ = gate output

$\underline{x}^n = (\underline{x}_i)_{i=0,1,\dots,n-1}$, where $\underline{x}_i \in \{0, 1\}$ are gate inputs.

$\underline{\pi}^n = (\underline{\pi}_i)_{i=0,1,\dots,n-1}$, 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.40.1, is

$$P(y = 1 | \underline{x}^n, \underline{\lambda}, \underline{\pi}^n) = 1 - (1 - \lambda) \prod_i [1 - \pi_i x_i] \quad (40.1)$$

$$P(y = 0 | \underline{x}^n, \underline{\lambda}, \underline{\pi}^n) = 1 - P(y = 1 | \underline{x}^n, \underline{\lambda}, \underline{\pi}^n) \quad (40.2)$$

Note that if $\lambda = 0$ and $\pi_i = 1$ for all i , then this becomes a deterministic OR-gate. Indeed,

$$P(y = 1 | \underline{x}^n, \lambda = 0, \underline{\pi}^n = 1^n) = 1 - \prod_i [1 - x_i] = \vee_{i=0}^{n-1} x_i , \quad (40.3)$$

so

$$P(y|x^n, \lambda = 0, \pi^n = 1^n) = \delta(y, \vee_{i=0}^{n-1} x_i) . \quad (40.4)$$

40.1 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

$$P(y = 1|x^n = e_i^n, \lambda = 0, \pi^n) = 1 - [1 - \pi_i] = \pi_i . \quad (40.5)$$

This gives an interpretation to the parameters π_i .

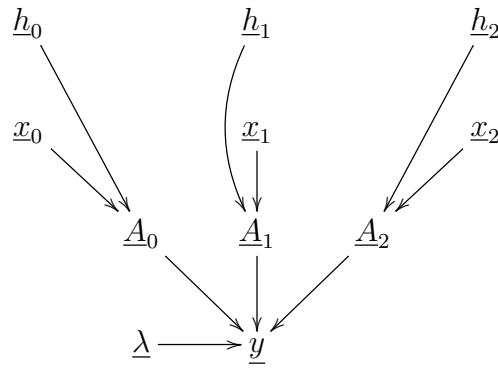


Figure 40.2: Fig.40.1 after replacing parameters $(\pi_i)_{i=0,1,2}$ by hidden nodes $(h_i)_{i=0,1,2}$.

2. Another way of interpreting the parameters π_i is to associate each of them with a hidden variable $h_i \in \{0, 1\}$ whose average equals π_i . More precisely, consider Fig.40.2.

Let $x_i, h_i, A_i, y \in \{0, 1\}$.

The TPMs, printed in blue, for the nodes of the bnet Fig.40.2, are as follows:

$$P(h_i) = \pi_i \delta(h_i, 1) + (1 - \pi_i) \delta(h_i, 0) \quad (40.6)$$

$$P(A_i|h_i, x_i) = \delta(A_i, h_i \wedge x_i) = \delta(A_i, h_i x_i) \quad (40.7)$$

$$P(y = 1|A^n) = 1 - (1 - \lambda) \wedge_{i=0}^{n-1} \bar{A}_i \quad (40.8)$$

$$= 1 - (1 - \lambda) \prod_i (1 - A_i) \quad (40.9)$$

$$P(y = 0|A^n) = 1 - P(y = 1|A^n) \quad (40.10)$$

Note that

$$P(y = 1|x^n, \lambda) = \sum_{h^n} \sum_{A^n} \left[1 - (1 - \lambda) \prod_i (1 - A_i) \right] [\prod_i \delta(A_i, h_i x_i)] P(h^n) \quad (40.11)$$

$$= E_{\underline{h}^n} \left[[1 - (1 - \lambda) \prod_i (1 - h_i x_i)] \right]. \quad (40.12)$$

But

$$E_{\underline{h}_i}[h_i x_i] = \sum_{h_i=0,1} P(h_i) h_i x_i = \pi_i x_i \quad (40.13)$$

so

$$P(y = 1|x^n, \lambda) = 1 - (1 - \lambda) \prod_i (1 - \pi_i x_i). \quad (40.14)$$

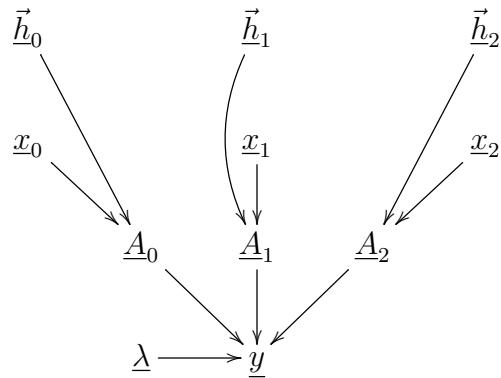


Figure 40.3: Fig.40.2 after replacing the hidden nodes $(\underline{h}_i)_{i=0,1,2}$ by vectors of samples $(\vec{h}_i)_{i=0,1,2}$.

3. Another way to interpret the parameters π_i is to associate each of them with a vector of samples \vec{h}_i whose average is π_i . More precisely, consider Fig.40.3.

Suppose $\underline{h}_i \in \{0, 1\}$ and define

$$P_{\underline{h}_i}(h_i) = \pi_i \delta(h_i, 1) + (1 - \pi_i) \delta(h_i, 0) . \quad (40.15)$$

Suppose $\vec{h}_i = (\underline{h}_i[\sigma])_{s=0,1,\dots,n_{sam}-1}$ 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{h}_i as follows:

$$\hat{P}_{\underline{h}_i}(h_i) = \frac{1}{n_{sam}} \sum_{\sigma=0}^{n_{sam}-1} \mathbb{1}(\underline{h}_i[\sigma] = h_i) . \quad (40.16)$$

Let $x_i, \underline{h}_i[\sigma], \underline{A}_i, y \in \{0, 1\}$.

The TPMs, printed in blue, for the nodes of the bnet Fig.40.3, are as follows:

$$P(\vec{h}_i) = \prod_{\sigma=0}^{n_{sam}-1} P_{\underline{h}_i}(\underline{h}_i[\sigma]) \quad (40.17)$$

$$P(A_i | \vec{h}_i, x_i) = \delta(A_i, \frac{1}{n_{sam}} \sum_{\sigma} \underline{h}_i[\sigma] \wedge x_i) \quad (40.18)$$

$$= \delta(A_i, \pi_i x_i) \quad (40.19)$$

$$P(y = 1 | A^n) = 1 - (1 - \lambda) \wedge_{i=0}^{n-1} \bar{A}_i \quad (40.20)$$

$$= 1 - (1 - \lambda) \prod_i (1 - A_i) \quad (40.21)$$

$$P(y = 0 | A^n) = 1 - P(y = 1 | A^n) \quad (40.22)$$

Note that

$$P(y = 1 | x^n, \lambda, \vec{h}^n) = \sum_{A^n} \left[1 - (1 - \lambda) \prod_i (1 - A_i) \right] \prod_i \delta(A_i, \pi_i x_i) \quad (40.23)$$

$$= 1 - (1 - \lambda) \prod_i (1 - \pi_i x_i) . \quad (40.24)$$

Chapter 41

Non-negative Matrix Factorization

Based on Ref.[82].

Given matrix V , factor it into product of two matrices

$$V = WH , \quad (41.1)$$

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)

41.1 Bnet interpretation

Express node \underline{v} as a chain of two nodes.

$$\underline{v} \longleftarrow \underline{a} \quad = \quad \underline{w} \longleftarrow \underline{h} \longleftarrow \underline{a}$$

Figure 41.1: Bnet interpretation of non-negative matrix factorization.

Node TPMs, printed in blue, for Fig.41.1.

$$P(\underline{v} = w | \underline{a}) = \frac{V_{w,a}}{\sum_w V_{w,a}} \quad (41.2)$$

$$P(w|h) = \frac{W_{w,h}}{\sum_w W_{w,h}} \quad (41.3)$$

$$P(h|a) = \frac{\sum_w W_{w,h}}{\sum_w V_{w,a}} H_{h,a} \quad (41.4)$$

41.2 Simplest recursive algorithm

Initialize: Choose nh . Choose $W^{(0)}$ and $H^{(0)}$ that have non-negative entries.

Update: For $n = 0, 1, \dots$, do

$$H_{i,j}^{(n+1)} \leftarrow H_{i,j}^{(n)} \frac{[(W^{(n)})^T V]_{i,j}}{[(W^{(n)})^T \underbrace{W^{(n)} H^{(n)}}_{\approx V}]_{i,j}} \quad (41.5)$$

and

$$W_{i,j}^{(n+1)} \leftarrow W_{i,j}^{(n)} \frac{[V(H^{(n+1)})^T]_{i,j}}{\underbrace{[W^{(n)} H^{(n+1)} (H^{(n+1)})^T]_{i,j}}_{\approx V}}. \quad (41.6)$$

After each step, record error defined by

$$\mathcal{E}^{(n)} = \| V - W^{(n)} H^{(n)} \|_2. \quad (41.7)$$

Using 2-norm, aka Frobenius matrix norm. Continue until reach acceptable error.

Can also use Kullback-Lieber divergence for error:

$$\mathcal{E} = \sum_a P(a) D_{KL}(P(\underline{v} = w|a) \parallel \sum_h P(w|h)P(h|a)), \quad (41.8)$$

for some arbitrary choice of prior $P(a)$. For example, can choose $P(a)$ uniform.

Chapter 42

Observationally Equivalent DAGs

This chapter is based on Chapter 1 of Ref.[27] and on a blog post by Bruno Gonçalves (Ref.[8]).

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

$$\mathcal{P}(G) = \{P : P \text{ is compatible with } G\} . \quad (42.1)$$

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

$$P(a|b)P(b) = P(a, b) = P(b|a)P(a) . \quad (42.2)$$

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 16 for definition of $DS(G)$.

Claim 26 *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.42.1 shows in red all the v-structures of a particular DAG.

Claim 27 *Observational Equivalence Theorem (by Verma and Pearl, 1990)*

Two DAGs are OE iff they have the same skeletons and the same v-structures.

42.1 Examples

The 3 DAGs in Fig.42.2 are OE. They form an equivalence class of OE DAGs that represent the same probability distribution. This equivalence class of DAGs can be

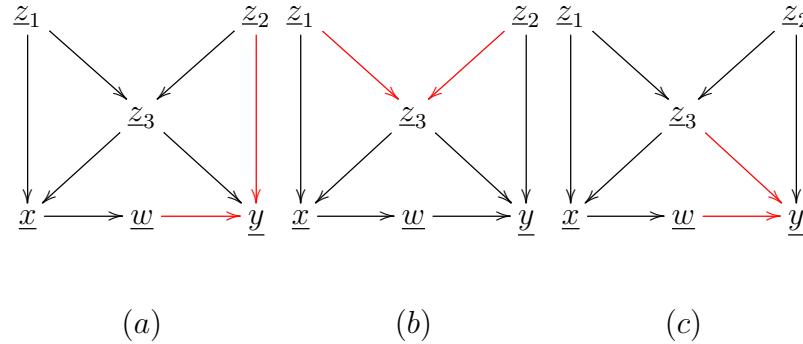


Figure 42.1: Example showing in red all v-structures of a particular DAG.

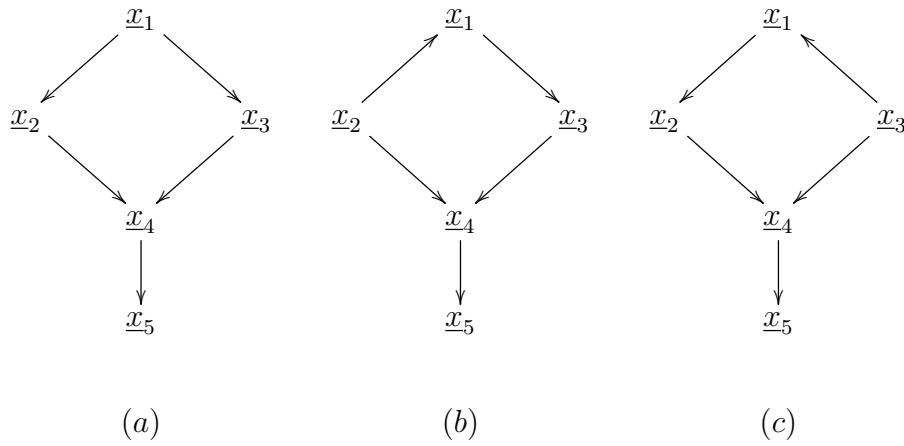


Figure 42.2: These 3 DAGs are observationally equivalent (OE).

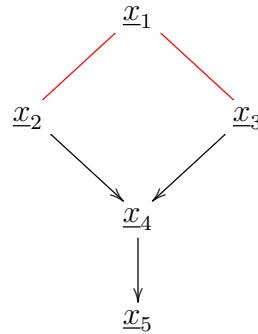


Figure 42.3: This partially directed graph represents the 3 DAGs in Fig.42.2.

represented by the partially directed graph Fig.42.3. 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.42.2. That is the low brow way of proving OE.
2. Use d-separation (see Chapter 16). Consider DAG (a) first. Rename the nodes as $\underline{\tau}_j$ with $j = 1, 2, \dots$ 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 topological 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.

$$P(x_1, x_2, x_3, x_4, x_5) = \underbrace{P(x_5|x_4)}_{x_3, x_2, x_1 \text{ omitted}} \underbrace{P(x_4|x_3, x_2)}_{x_1 \text{ omitted}} \underbrace{P(x_3|x_1)}_{x_2 \text{ omitted}} P(x_2|x_1)P(x_1) \quad (42.3)$$

The observations of which parents were omitted can be stated in d-separation lingo as the following 3 orthogonality relations:¹

$$\underline{x}_3 \perp_P \underline{x}_2 \mid \underline{x}_1 \quad (42.4a)$$

$$\underline{x}_4 \perp_P \underline{x}_1 \mid \underline{x}_2, \underline{x}_3 \quad (42.4b)$$

$$\underline{x}_5 \perp_P (\underline{x}_1, \underline{x}_2, \underline{x}_3) \mid \underline{x}_4 . \quad (42.4c)$$

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.42.2 are OE.

Note that Eqs.(42.4) 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 $\underline{x}_2 \rightarrow \underline{x}_4 \leftarrow \underline{x}_3$.

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.

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 43

Potential Outcomes

This chapter is based on Ref.[3], a book by Stephen Cunningham entitled “Causal inference: the mixtape”.

The theory of potential outcomes (PO) was for the most part invented in a seminal 1974 paper by Donald B. Rubin. Rubin has also made important extensions to PO theory since 1974. However, he refuses to use Pearl’s causal DAGs to discuss PO theory. Pearl has shown that PO theory can be substantially clarified and extended by using the language of causal DAGs. The d-separation theorem that we discuss in Chapter 16 is especially useful in this regard.

In this chapter, we stress the connection of PO theory to bnets, and, in particular, to the do and imagine operators defined in Chapter 10. Hence, before reading this chapter, the reader is expected to have at least skimmed Chapter 10, so that he/she understands the definition of do and imagine operators.

σ	d^σ	y^σ	$y^\sigma(0)$	$y^\sigma(1)$
Edith	0	5	5	.
Frank	0	7	7	.
George	0	8	8	.
Hank	0	10	10	.
Andy	1	10	.	10
Ben	1	5	.	5
Chad	1	16	.	16
Daniel	1	3	.	3

Table 43.1: PO dataset describing whether individual σ took a treatment dose ($d^\sigma = 1$) or didn’t ($d^\sigma = 0$). The treatment outcome is measured by the real number y^σ .

Suppose a **population of individuals** $\sigma = 0, 1, 2, \dots, nsam - 1$ is given ($d^\sigma = 1$) or not given ($d^\sigma = 0$) a **treatment discrete drug dose** d^σ , and that the **treatment outcome (i.e., response)** is measured by a real number y^σ . Table 43.1 gives a possible **PO dataset** for this scenario. As you can see from that table, each individual either takes a drug dose or doesn’t, but not both. PO theory can be

viewed as a **missing data (MD) problem**. MD problems are discussed in Chapter 36. However, the PO MD problem is much more specialized than the generic MD problems discussed in Chapter 36. In the PO MD problem, we can fill in the blank cells by matching each individual that took the drug with another *similar* individual that didn't. We will have much more to say about this matching strategy later in this chapter.

One can define similar individuals as individuals that have the same value for nx features $x^\sigma = (x_i^\sigma)_{i=0,1,\dots,nx-1}$. One can add to Table 43.1 nx extra columns giving the value of the feature vector x^σ for each individual. Members of a population with the same x^σ are referred to as a **subpopulation or stratum** (ie., layer).

In a **randomized clinical trial (RCT)**¹, the effect of the variable x^σ on the value of d^σ is eliminated by randomizing the population and therefore making the effect of x^σ average out to zero. However, there are many situations in which carrying out an RCT is not possible. PO theory is a way of predicting the result of an RCT in situations where doing a real RCT is not physically possible.

In this chapter, x^σ will be called the confounders. Implicit throughout this chapter is the assumption that there are **no unmeasured confounders**. Because if there are some unmeasured confounders, those can send secret messages that influence the value that d^σ takes. This would ruin the predictions of someone trying to predict the results of an RCT without being privy to those secret messages. When there are **some unmeasured confounders**, it might still be possible to predict the effect of an RCT. This might be possible using instrumental variables. See Chapter 26 for a discussion of **instrumental variables**.

43.1 G and G_{den} , bnets, the starting point bnets

In this chapter, we will abbreviate $\underline{X}[\sigma] = \underline{X}^\sigma$ for $X \in \{d, x, y\}$ and for $\sigma = \{0, 1, 2, \dots, nsam - 1\}$.

For each individual (aka unit, sample) $\sigma = 0, 1, 2, \dots, nsam - 1$, let:

$d^\sigma \in \{0, 1\}$: treatment discrete drug dose, 1 if treated and 0 if untreated

$y^\sigma \in \mathbb{R}$: treatment potential outcome

\underline{x}^σ : column vector of treatment confounders (aka covariates, because they are often used as covariates (i.e., independent variables) in linear regression.)

Consider bnets G and G_{den} in Fig.43.1. G reflects the language used in Ref.[3] to discuss PO theory. And G_{den} reflects the language that Judea Pearl prefers to use to discuss PO theory. Both languages are equivalent. To go from one language to the other, one need only perform the following swaps, where \underline{u} is the external noise of the DEN bnet.

$\underline{X}^\sigma \leftrightarrow \underline{X}(\underline{u})$ for $X \in \{d, x, y\}$.

¹The term **A/B test** is often used to mean a RCT where A and B are the treated and control groups. However, sometimes the term is used to refer to an experiment that conditions on confounders, which violates the definition of a RCT, and is the same as a PO test.

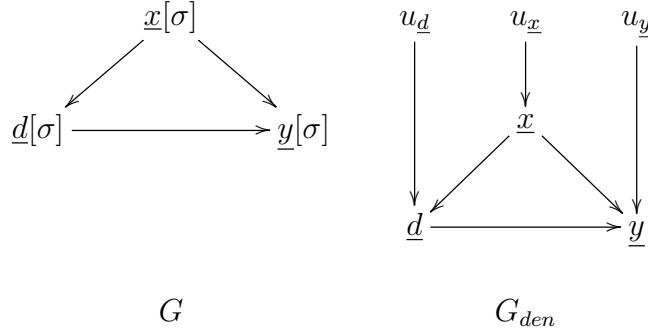


Figure 43.1: Bnets G and G_{den} are our starting point in discussing PO theory. G is for a single individual σ of the population. Bnet G_{den} is the DEN counterpart to G . DEN (Deterministic with External Noise) bnets are discussed in Chapter 31.

$$P(\sigma) = \frac{1}{nsam} \leftrightarrow P(u)$$

$$\sum_{\sigma} P(\sigma)(\cdot) \leftrightarrow \sum_u P(u)(\cdot)$$

The TPMs, printed in blue, for the bnet G in Fig.43.1, are as follows:

$$P(x^{\sigma}) = P_{\underline{x}}(x^{\sigma}) \quad (43.1)$$

$$P(d^{\sigma}|x^{\sigma}) = P_{d|\underline{x}}(d^{\sigma}|x^{\sigma}) \quad (43.2)$$

$$P(y^{\sigma}|x^{\sigma}, d^{\sigma}) = P_{y|\underline{x}, d}(y^{\sigma}|x^{\sigma}, d^{\sigma}) \quad (43.3)$$

Now let:

$\underline{d} \in \{0, 1\}$: treatment discrete drug dose, 1 if treated and 0 if untreated

$\underline{y} \in \mathbb{R}$: treatment potential outcome

\underline{x} : column vector of treatment confounders (aka covariates)

$\underline{u} = (\underline{u}_d, \underline{u}_x, \underline{u}_y)$: external noise

The TPMs, printed in blue, for the bnet G_{den} in Fig.43.1, are as follows:

$$P(x|u_{\underline{x}}) = \mathbb{1}(x = u_{\underline{x}}) \quad (43.4)$$

$$P(d|x, u_{\underline{d}}) = \mathbb{1}(d = f_d(x, u_{\underline{d}})) \quad (43.5)$$

$$P(y|d, x, u_{\underline{y}}) = \mathbb{1}(y = f_y(d, x, u_{\underline{y}})) \quad (43.6)$$

If we linearize $f_{\underline{y}}$ in Eq.(43.6), we get

$$\underline{y} = \delta \underline{d} + \beta \underline{x} + \underline{u}_y , \quad (43.7)$$

where $\delta, \beta \in \mathbb{R}$. Assuming that $\underline{x}, \underline{y} \in \mathbb{R}$ and $\underline{d} \in \{0, 1\}$, Eq.(43.7) can be plotted. The resulting plot is given in Fig.43.2. This plot is a very special case of the PO problem, but it gives a crude idea of the “effects” $\delta = y(1) - y(0)$ that PO theory gives estimates for. Any individual participating in the experiment experiences either $y(1)$ or $y(0)$, but not both.

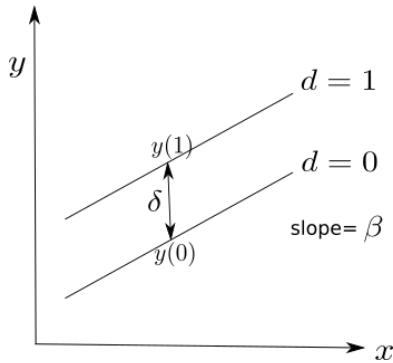


Figure 43.2: Plot of Eq.(43.7)

43.2 G_{do+} bnet

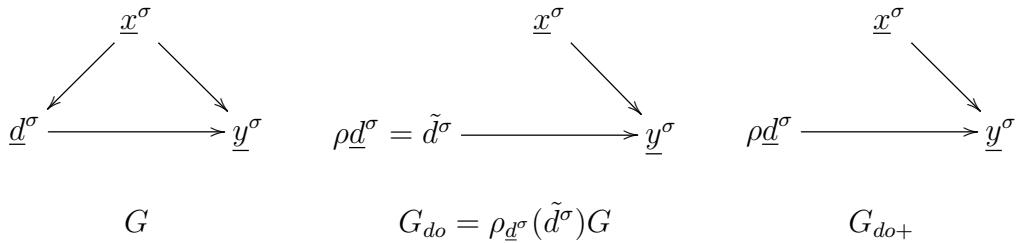


Figure 43.3: Bnet $G_{do} = \rho_{d^{\sigma}}(\tilde{d}^{\sigma})G$ is obtained by applying the do operator to node d^{σ} of bnet G . Bnet G_{do+} is obtained by adding a prior probability distribution $P(\tilde{d}^{\sigma})$ to node ρd^{σ} of bnet G_{do} .

Fig.43.3 shows how bnet G_{do} is obtained by applying the do operator to bnet G , and how bnet G_{do+} is obtained by adding a prior probability distribution to one

of the nodes of G_{do} . In bnet G_{do} , node \underline{d}^σ has been stripped of all outside influences and fixed to a specific state \tilde{d}^σ . This is what an RCT does.

The TPMs, printed in blue, for the bnets G_{do} and G_{do+} , are as follows. Note that the TPMs for bnets G_{do} and G_{do+} are defined in terms of the TPMs of bnet G .

$$P(x^\sigma) = P_{\underline{x}}(x^\sigma) \quad (43.8)$$

$$P_{\rho_d}(d) = \sum_x P_{\underline{d}|\underline{x}}(d|x) P_{\underline{x}}(x) \quad (43.9)$$

$$P(\tilde{d}^\sigma) = \begin{cases} \delta(\tilde{d}^\sigma, (\tilde{d}^\sigma)') & \text{for } G_{do} \\ P_{\rho_d}(\tilde{d}^\sigma) & \text{for } G_{do+} \end{cases} \quad (43.10)$$

$$P(y^\sigma|x^\sigma, \tilde{d}^\sigma) = P_{\underline{y}|\underline{x}, \underline{d}}(y^\sigma|x^\sigma, \tilde{d}^\sigma) \quad (43.11)$$

It is convenient to define the following expected values of y^σ in terms of the TPMs of bnet G_{do+} :

$$\mathcal{Y}_{|\tilde{d},x} = E_{\sigma|\tilde{d},x}[\underline{y}^\sigma] \rightarrow E_{\underline{y}|\tilde{d},x}[\underline{y}] = \sum_y y P(y|\tilde{d},x) \quad (43.12)$$

$$\mathcal{Y}_{|\tilde{d}} = E_{\sigma|\tilde{d}}[\underline{y}^\sigma] \rightarrow E_{y|\tilde{d}}[\underline{y}] = \sum_x \mathcal{Y}_{|\tilde{d},x} P(x) \quad (43.13)$$

$$\mathcal{Y}_{|x} = E_{\sigma|x}[\underline{y}^\sigma] \rightarrow E_{y|x}[\underline{y}] = \sum_{\tilde{d}} \mathcal{Y}_{|\tilde{d},x} P(\tilde{d}) \quad (43.14)$$

$$\mathcal{Y} = E_\sigma[\underline{y}^\sigma] \rightarrow E_y[\underline{y}] = \sum_{\tilde{d},x} \mathcal{Y}_{|\tilde{d},x} P_{\underline{d}|\underline{x}}(\tilde{d}|x) P(x) \quad (43.15)$$

43.3 G_{im+} bnet

Fig.43.4 shows how bnet G_{im} is obtained by applying an imagine operator to bnet G , and how bnet G_{im+} is obtained by adding a prior probability distribution to one of the nodes of G_{im} . $\underline{d} \in \{0, 1\}$ represents the dose that a patient is told to take by a doctor, and $\tilde{d} \in \{0, 1\}$ represents the dose he actually takes. If $\underline{d} = \tilde{d}$, the patient is compliant, and if $\underline{d} \neq \tilde{d}$, he is non-compliant.

The TPMs, printed in blue, for the nodes of bnets G_{im} and G_{im+} , are as follows. Note that the TPMs for bnets G_{im} and G_{im+} are defined in terms of the

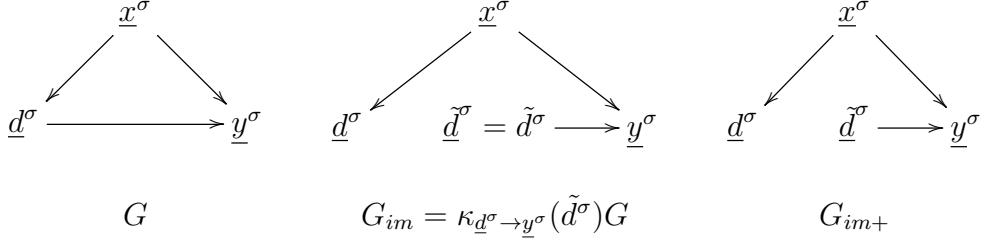


Figure 43.4: Bnet $G_{im} = \kappa_{\underline{d}^\sigma \rightarrow \underline{y}^\sigma}(\tilde{d}^\sigma)G$ is obtained by applying the imagine operator to arrow $\underline{d}^\sigma \rightarrow \underline{y}^\sigma$ of bnet G . Bnet G_{im+} is obtained by adding a prior probability distribution $P(\tilde{d}^\sigma)$ to node \tilde{d}^σ of bnet G_{im} .

TPMs of bnet G . Note that the prior $P(\tilde{d})$ is not arbitrary; it's calculated from the TPMs of bnet G .

$$P(x^\sigma) = P_{\underline{x}}(x^\sigma) \quad (43.16)$$

$$P(d^\sigma|x^\sigma) = P_{\underline{d}|\underline{x}}(d^\sigma|x^\sigma) \quad (43.17)$$

$$\pi_{\tilde{d}} = P(\tilde{d}) = \sum_x P_{\underline{d}|\underline{x}}(\tilde{d}|x)P_{\underline{x}}(x) \quad (43.18)$$

$$P(\tilde{d}^\sigma) = \begin{cases} \delta(\tilde{d}^\sigma, (\tilde{d}^\sigma)') & \text{for } G_{im} \\ \pi_{\tilde{d}^\sigma} & \text{for } G_{im+} \end{cases} \quad (43.19)$$

$$P(y^\sigma|x^\sigma, \tilde{d}^\sigma) = P_{\underline{y}|\underline{x}, \underline{d}}(y^\sigma|x^\sigma, \tilde{d}^\sigma) \quad (43.20)$$

43.4 G_{im+} bnet with nodes $y^\sigma(0), y^\sigma(1)$ added to it.

Consider Fig.43.5, which was obtained by adding two new nodes $y^\sigma(0)$ and $y^\sigma(1)$ to bnets G_{im} and G_{im+} in Fig.43.4. The TPMs, printed in blue, for bnets G_{im} and G_{im+} , are as follows. Note that we define them in terms of the TPMs for bnet G .

$$P(x^\sigma) = P_{\underline{x}}(x^\sigma) \quad (43.21)$$

$$P(d^\sigma|x^\sigma) = P_{\underline{d}|\underline{x}}(d^\sigma|x^\sigma) \quad (43.22)$$

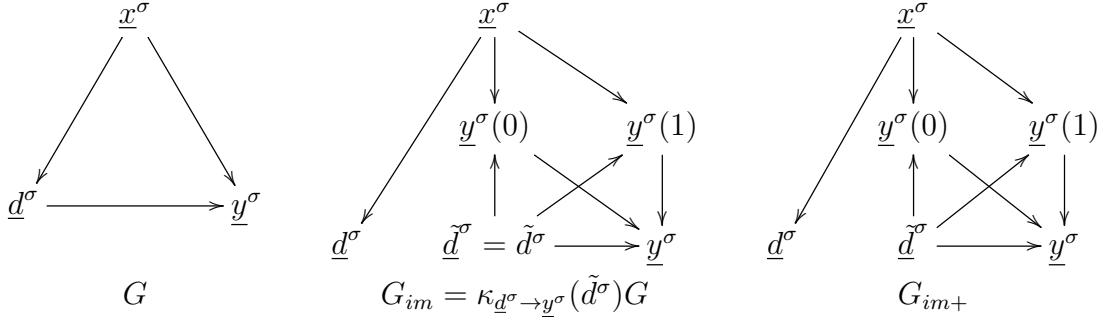


Figure 43.5: Fig.43.4 with two new nodes $\underline{y}^{\sigma}(0)$ and $\underline{y}^{\sigma}(1)$ added to bnets G_{im} and G_{im+} .

$$\pi_{\tilde{d}} = P(\tilde{d}) = \sum_x P_{\underline{d}|x}(\tilde{d}|x)P_x(x) \quad (43.23)$$

$$P(\tilde{d}^{\sigma}) = \begin{cases} \delta(\tilde{d}^{\sigma}, (\tilde{d}^{\sigma})') & \text{for } G_{im} \\ \pi_{\tilde{d}^{\sigma}} & \text{for } G_{im+} \end{cases} \quad (43.24)$$

$$P(y^{\sigma}(0)|\tilde{d}^{\sigma}, x^{\sigma}) = P_{\underline{y}(0)|\tilde{d},x}(y^{\sigma}(0)|\tilde{d}^{\sigma}, x^{\sigma}) \quad (43.25)$$

$$P(y^{\sigma}(1)|\tilde{d}^{\sigma}, x^{\sigma}) = P_{\underline{y}(1)|\tilde{d},x}(y^{\sigma}(1)|\tilde{d}^{\sigma}, x^{\sigma}) \quad (43.26)$$

$$P(y^{\sigma}|y^{\sigma}(0), y^{\sigma}(1), \tilde{d}^{\sigma}) = \mathbb{1}(y^{\sigma} = \tilde{d}^{\sigma}y^{\sigma}(1) + (1 - \tilde{d}^{\sigma})y^{\sigma}(0)) \quad (43.27)$$

$$= \mathbb{1}(y^{\sigma} = y^{\sigma}(\tilde{d}^{\sigma})) \quad (43.28)$$

For this bnet, the following is true:

$$P(y^{\sigma}|\tilde{d}^{\sigma}, x^{\sigma}) = \sum_{y^{\sigma}(0)} \sum_{y^{\sigma}(1)} \mathbb{1}(y^{\sigma} = y^{\sigma}(\tilde{d}^{\sigma})) P(y^{\sigma}(0)|\tilde{d}^{\sigma}, x^{\sigma}) P(y^{\sigma}(1)|\tilde{d}^{\sigma}, x^{\sigma}) \quad (43.29)$$

$$= \begin{cases} P_{\underline{y}(0)|\tilde{d},x}(y^{\sigma}|\tilde{d}^{\sigma}, x^{\sigma}) & \text{if } \tilde{d}^{\sigma} = 0 \\ P_{\underline{y}(1)|\tilde{d},x}(y^{\sigma}|\tilde{d}^{\sigma}, x^{\sigma}) & \text{if } \tilde{d}^{\sigma} = 1 \end{cases} \quad (43.30)$$

Note that $P_{\underline{y}(0)|\tilde{d},x}$ and $P_{\underline{y}(1)|\tilde{d},x}$ are possibly different functions and that $P_{\underline{y}|\tilde{d},x}$ is defined in terms of both of them. Eq.43.30 implies that for this bnet,

$$\underline{y}^\sigma = \mathbb{1}(\tilde{d}^\sigma = 1)\underline{y}(1) + \mathbb{1}(\tilde{d}^\sigma = 0)\underline{y}(0) \quad (43.31)$$

$$= \tilde{d}^\sigma \underline{y}(1) + (1 - \tilde{d}^\sigma) \underline{y}(0) \quad (43.32)$$

$$= \underline{y}^\sigma(\tilde{d}^\sigma) \quad (43.33)$$

These are all different ways of saying the same thing.

It is convenient to define the following expected values of \underline{y}^σ in terms of the TPMs of bnet G_{im+} :

$$\mathcal{Y}_{d|\tilde{d},x} = E_{\sigma|\tilde{d},x}[\underline{y}^\sigma(d)] \rightarrow E_{\underline{y}|\tilde{d},x}[\underline{y}(d)] = \sum_y P(y|\tilde{d},x) y(\tilde{d}) \quad (43.34)$$

$$\mathcal{Y}_{d|\tilde{d}} = E_{\sigma|\tilde{d}}[\underline{y}^\sigma(d)] \rightarrow E_{\underline{y}|\tilde{d}}[\underline{y}(d)] = \sum_x \mathcal{Y}_{d|\tilde{d},x} P(x) \quad (43.35)$$

$$\mathcal{Y}_{d|x} = E_{\sigma|x}[\underline{y}^\sigma(d)] \rightarrow E_{\underline{y}|x}[\underline{y}(d)] = \sum_{\tilde{d}} \mathcal{Y}_{d|\tilde{d},x} P(\tilde{d}) \quad (43.36)$$

$$\mathcal{Y}_d = E_\sigma[\underline{y}^\sigma(d)] \rightarrow E_{\underline{y}}[\underline{y}(d)] = \sum_{\tilde{d},x} \mathcal{Y}_{d|\tilde{d},x} P_{\tilde{d}|x}(\tilde{d}|x) P(x) \quad (43.37)$$

$\mathcal{Y}_{0|0}, \mathcal{Y}_{1|1}$ are said to be **factual** (indicating compliant patients) whereas $\mathcal{Y}_{0|1}, \mathcal{Y}_{1|0}$ are said to be **counterfactual** (indicating non-compliant patients).

43.5 Conditional Independence Assumption

The **Conditional Independence Assumption** (CIA) is said to hold if

$$(\underline{y}^\sigma(0), \underline{y}^\sigma(1), \underline{y}^\sigma, \tilde{d}^\sigma) \perp_P \underline{d}^\sigma | \underline{x}^\sigma. \quad (43.38)$$

This is satisfied by G_{im} . To prove this, check that

$$(\underline{y}^\sigma(0), \underline{y}^\sigma(1), \underline{y}^\sigma, \tilde{d}^\sigma) \perp_{G_{im}} \underline{d}^\sigma | \underline{x}^\sigma \quad (43.39)$$

and then invoke the d-separation theorem (see Chapter 16).

Note that the following are also true

$$(\underline{y}^\sigma(0), \underline{y}^\sigma(1)) \perp_{G_{im}} \tilde{d}^\sigma | \underline{x}^\sigma \quad (43.40)$$

$$(\underline{y}^\sigma(0), \underline{y}^\sigma(1)) \perp_{G_{im}} \underline{d}^\sigma \quad (43.41)$$

by the d-separation theorem, because \underline{y}^σ acts as a collider on all paths from $\underline{y}^\sigma(d)$ to $\tilde{\underline{d}}^\sigma$ for $d \in \{0, 1\}$. However,

$$\underline{y}^\sigma \perp_{G_{im}} \tilde{\underline{d}}^\sigma | \underline{x}^\sigma \text{ is FALSE.} \quad (43.42)$$

Note that even though CIA means $\underline{d}^\sigma \perp_{G_{im}} \underline{d}^\sigma | \underline{x}^\sigma$, this does not mean that $\mathcal{Y}_{d|\tilde{d},x} = \mathcal{Y}_{d|x}$. The reason is that $\mathcal{Y}_{d|\tilde{d},x} = E_{|\tilde{d}^\sigma=\tilde{d},x}[\underline{y}^\sigma(d)]$ so $\mathcal{Y}_{d|\tilde{d},x} = \mathcal{Y}_{d|x}$ if $\underline{y}^\sigma(d) \perp_{G_{im}} \tilde{\underline{d}}^\sigma | \underline{x}^\sigma$, which is false. It is possible for $\mathcal{Y}_{d|\tilde{d},x} = \mathcal{Y}_{d|x}$ to be true. We discuss that situation in Section 43.9.

43.6 $\mathcal{Y}_{|\tilde{d},x}$ and G_{do}

Note that $\mathcal{Y}_{d|x}$ and $\mathcal{Y}_{|\tilde{d},x}$ are not the same thing.

$$\mathcal{Y}_{d|x} = E_{|x}[\underline{y}(d)] \quad (43.43)$$

whereas

$$\mathcal{Y}_{|\tilde{d},x} = E_{|\tilde{d},x}[\underline{y}] . \quad (43.44)$$

Claim 28

$$\mathcal{Y}_{|\tilde{d},x} = \mathcal{Y}_{\tilde{d}|\tilde{d},x} \quad (43.45)$$

proof:

$$\mathcal{Y}_{\tilde{d}|\tilde{d},x} = \tilde{d}\mathcal{Y}_{1|\tilde{d},x} + (1-\tilde{d})\mathcal{Y}_{0|\tilde{d},x} \quad (43.46)$$

$$= \tilde{d}E_{|\tilde{d},x}[\underline{y}(1)] + (1-\tilde{d})E_{|\tilde{d},x}[\underline{y}(0)] \quad (43.47)$$

$$= E_{|\tilde{d},x}[\tilde{d}\underline{y}(1)] + (1-\tilde{d})\underline{y}(0)] \quad (43.48)$$

$$= E_{|\tilde{d},x}[\underline{y}] \quad (43.49)$$

$$= \mathcal{Y}_{|\tilde{d},x} \quad (43.50)$$

QED

$\mathcal{Y}_{|\tilde{d},x}$ is connected to the do operator as follows.

$$\mathcal{Y}_{|\tilde{d},x} = \sum_y y P(\underline{y} = y | \rho \underline{d} = \tilde{d}, \underline{x} = x) , \quad (43.51)$$

where

$$P(\underline{y} = y | \rho \underline{d} = \tilde{d}, \underline{x} = x) = P(y | \tilde{d}, x) . \quad (43.52)$$

In particular, when \underline{y} is binary (i.e., $\underline{y} \in \{0, 1\}$), Eq.(43.51) becomes

$$\mathcal{Y}_{|\tilde{d},x} = P(\underline{y} = 1 | \rho \underline{d} = \tilde{d}, \underline{x} = x) . \quad (43.53)$$

43.7 Translation Dictionary

In standard PO notation	In our notation (for G_{im} or G_{im+})
i , individual (i.e., unit, sample) index	σ
$D_i = d_i$, treatment dose	$\underline{d}^\sigma = d^\sigma$
$Y_i = y_i$, treatment outcome	$\underline{y}^\sigma = y^\sigma$
$X_i = x_i$, treatment confounders	$\underline{x}^\sigma = x^\sigma$
$E[Y_i(d)]$	$E_\sigma[\underline{y}^\sigma(d)] = \mathcal{Y}_d$
$E[Y_i D_i = \tilde{d}]$	$E_{\sigma \tilde{d}}[\underline{y}^\sigma] = \mathcal{Y}_{ \tilde{d}}$
$E[Y_i(d) D_i = \tilde{d}]$	$E_{\sigma \tilde{d}}[\underline{y}^\sigma(d)] = \mathcal{Y}_{d \tilde{d}}$
$E[Y_i(d) D_i = \tilde{d}, X_i = x]$	$E_{\sigma \tilde{d},x}[\underline{y}^\sigma(d)] = \mathcal{Y}_{d \tilde{d},x}$

Table 43.2: Dictionary for translating from standard PO notation of Ref.[3] to our notation.

Table 43.2 gives a dictionary for translating from the standard PO notation of Ref.[3] to our notation. $d, \tilde{d} \in \{0, 1\}$. I find the standard PO notation confusing because it often uses D_i to represent two different nodes, \underline{d}^σ and $\tilde{\underline{d}}^\sigma$ in G_{im+} . This confusion becomes particularly distressing when we are told in PO notation that

$$Y_i(d) = dY_i(1) + (1 - d)Y_i(0), \quad (43.54)$$

and

$$E \left[Y_i(d) | D_i = \tilde{d}, X_i = x \right] = \mathcal{Y}_{d|\tilde{d},x}. \quad (43.55)$$

In our notation, this is saying that

$$\underline{y}^\sigma(d) = d\underline{y}^\sigma(1) + (1 - d)\underline{y}^\sigma(0), \quad (43.56)$$

and

$$E_{\sigma|\underline{d}^\sigma = \tilde{d}, \underline{x}^\sigma = x}[\underline{y}^\sigma(d)] = \mathcal{Y}_{d|\tilde{d},x}. \quad (43.57)$$

43.8 $\mathcal{Y}_{d|\tilde{d}}$ differences (aka treatment effects)

It is convenient to define the following **treatment effects**. See Fig.43.6. Note that we use the word “**effect**” to refer to a difference of two $\mathcal{Y}_{d|\tilde{d}}$.

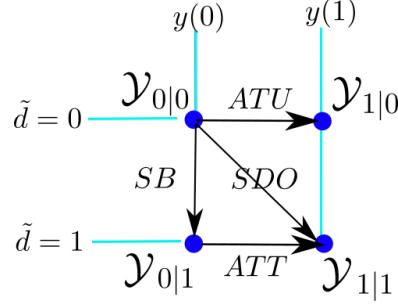


Figure 43.6: Different treatment effects. An effect is a difference of two $\mathcal{Y}_{d|\tilde{d}}$.

- average controlled causal effect (ACE), used when doing an RCT.

$$\textcolor{red}{ACE} = \mathcal{Y}_1 - \mathcal{Y}_0 = \mathcal{Y}_{1|1} - \mathcal{Y}_{0|0} = SDO \quad (43.58)$$

- average treatment effect² (ATE).

$$\textcolor{red}{ATE} = \mathcal{Y}_1 - \mathcal{Y}_0 = \delta \quad (43.59)$$

- average treatment effect of the treated (ATT)

$$\textcolor{red}{ATT} = \mathcal{Y}_{1|1} - \mathcal{Y}_{0|1} \quad (43.60)$$

- average treatment effect of the untreated (ATU)

$$\textcolor{red}{ATU} = \mathcal{Y}_{1|0} - \mathcal{Y}_{0|0} \quad (43.61)$$

- selection bias (SB)

$$\textcolor{red}{SB} = \mathcal{Y}_{0|1} - \mathcal{Y}_{0|0} \quad (43.62)$$

- simple difference in outcomes (SDO)

$$\textcolor{red}{SDO} = \mathcal{Y}_{1|1} - \mathcal{Y}_{0|0} \quad (43.63)$$

Note that some of these effects are linearly related

$$\underbrace{\mathcal{Y}_1 - \mathcal{Y}_0}_{ATE} = \underbrace{(\mathcal{Y}_{1|1} - \mathcal{Y}_{0|1})}_{ATT} \pi_1 + \underbrace{(\mathcal{Y}_{1|0} - \mathcal{Y}_{0|0})}_{ATU} \pi_0 \quad (43.64)$$

$$\underbrace{\mathcal{Y}_{1|1} - \mathcal{Y}_{0|0}}_{SDO} = \underbrace{(\mathcal{Y}_{1|1} - \mathcal{Y}_{0|1})}_{ATT} + \underbrace{\mathcal{Y}_{0|1} - \mathcal{Y}_{0|0}}_{SB} \quad (43.65)$$

² Note that effects in which \tilde{d} varies are called “controlled”, whereas those in which d varies instead, are called simply “treatments”. y is averaged over in both cases.

$$\underbrace{\mathcal{Y}_{1|1} - \mathcal{Y}_{0|0}}_{SDO} = \underbrace{(\mathcal{Y}_{1|1} - \mathcal{Y}_{0|1})\pi_1 + (\mathcal{Y}_{1|0} - \mathcal{Y}_{0|0})\pi_0}_{ATE} \quad (43.66)$$

$$+ \underbrace{\mathcal{Y}_{0|1} - \mathcal{Y}_{0|0}}_{SB} \quad (43.67)$$

$$+ \underbrace{(\mathcal{Y}_{1|1} - \mathcal{Y}_{0|1})}_{ATT} \pi_0 \quad (43.68)$$

$$- \underbrace{(\mathcal{Y}_{1|0} - \mathcal{Y}_{0|0})}_{ATU} \pi_0 \quad (43.69)$$

Let $\mathcal{E} \in \{ACE, ATE, ATT, ATU, SDO, SB\}$. \mathcal{E} can be defined for a fixed stratum x by replacing $\mathcal{Y}_{d|\tilde{d}}$ with $\mathcal{Y}_{d|\tilde{d},x}$. We will denote such an extension by $\mathcal{E}|_x$, or, sometimes, simply by \mathcal{E} . $ATE|_x$ is sometimes called the **Conditional Average Treatment Effect (CATE)**. We will use the term ATE to refer to CATE too.³

43.9 Zero ACE, $\mathcal{Y}_{1|0} = \mathcal{Y}_1$

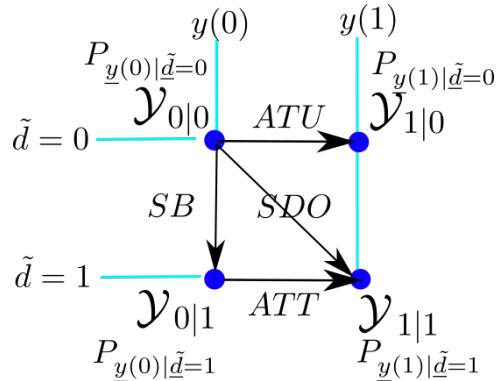


Figure 43.7: Figure 43.6 with added information about probability distributions used to obtain each expected value $\mathcal{Y}_{d|\tilde{d}}$.

$SDO = 0$ is the hypothesis tested by a Randomized Control Trial (RCT). But some people test for $ATE = 0$ instead.

1. Is it possible for $SDO = 0$ but $ATE \neq 0$ or vice versa, and what is going on when this is true?
2. What is going on when two treatment effects are equal; for instance, when $ATT = ATU$?

³Careful: We define $ATE|_x = ATT|_x P_{\underline{d}|x}(1|x) + ATU|_x P_{\underline{d}|x}(0|x)$. Therefore, $ATE|_x \neq ATT|_x P_{\underline{d}}(1) + ATU|_x P_{\underline{d}}(0)$.

3. When is $\mathcal{Y}_{1|0} = \mathcal{Y}_1$, and what is going on when this is true?

Fig.43.7 gives some intuition about what is going on when any of these things happen.

Recall that each expected value \mathcal{Y} has a probability distribution P .

$$\mathcal{Y}_{d|\tilde{d},x} = \sum_y y P_{\underline{y}(d)|\tilde{d},\underline{x}}(y|\tilde{d},x) \quad (43.70)$$

for $d, \tilde{d} \in \{0, 1\}$. Fig.43.7 reminds us of which P is used to generate each \mathcal{Y} . From this figure, we see that

1. A sufficient condition for $SDO = 0$ is that $P_{\underline{y}(1)|\tilde{d}=1,\underline{x}} = P_{\underline{y}(0)|\tilde{d}=0,\underline{x}}$. Since $ATE = 0$ iff $ATT = ATU = 0$, a sufficient condition for $ATE = 0$ is that $P_{\underline{y}(1)|\tilde{d}=0,\underline{x}} = P_{\underline{y}(0)|\tilde{d}=0,\underline{x}}$ and $P_{\underline{y}(1)|\tilde{d}=1,\underline{x}} = P_{\underline{y}(0)|\tilde{d}=1,\underline{x}}$.
2. A sufficient condition for $ATT = ATU$ is that $P_{\underline{y}(1)|\tilde{d}=0,x} - P_{\underline{y}(0)|\tilde{d}=0,x}$ equals $P_{\underline{y}(1)|\tilde{d}=1,x} - P_{\underline{y}(0)|\tilde{d}=1,x}$.
3. If we assume that all $y^\sigma > 0$, $\mathcal{Y}_{1|0} = \mathcal{Y}_1$ iff $P_{\underline{y}(1)|\tilde{d}=0,x} = P_{\underline{y}(1)|x}$.

$SDO = 0$ depends on two corners of the square whereas $\mathcal{Y}_{1|0} = \mathcal{Y}_1$ depends on just one corner.

43.10 (SDO, ATE) space

If we substitute $y^\sigma \rightarrow y^\sigma(\tilde{d}^\sigma)$ and $y^{s(\sigma)} \rightarrow y^\sigma(!\tilde{d}^\sigma)$, where $!0 = 1$ and $!1 = 0$, into the estimator Eq.(43.81) for ATE and the estimator Eq.(43.87) for SDO , we get

$$\widehat{ATE} = \frac{1}{N_x} \sum_{\sigma \in A_x} (2\tilde{d}^\sigma - 1)[y^\sigma(\tilde{d}^\sigma) - y^\sigma(!\tilde{d}^\sigma)] \quad (43.71)$$

$$= \frac{1}{N_x} \sum_{\sigma \in A_x} [y^\sigma(1) - y^\sigma(0)] \quad (43.72)$$

and

$$\widehat{SDO} = \frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma y^\sigma(\tilde{d}^\sigma) - \frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) y^\sigma(\tilde{d}^\sigma) \quad (43.73)$$

$$= \frac{1}{N_{1,x}} \sum_{\sigma \in A_{1,x}} y^\sigma(1) - \frac{1}{N_{0,x}} \sum_{\sigma \in A_{0,x}} y^\sigma(0). \quad (43.74)$$

$SDO = 0$ is the hypothesis tested by a Randomized Clinical Trial (RTC).

Suppose that the treatment outcome y^σ has only two possible values, 0 and 1. Then, $-1 \leq ATE \leq 1$ and $-1 \leq SDO \leq 1$. But does $ATE = 0$ imply $SDO = 0$ or vice versa? Next, we answer that question and more by finding the region of accessibility in the (SDO, ATE) plane, assuming $y^\sigma \in \{0, 1\}$.

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	1	0
2	0	1	0
3	0	1	0
4	1	1	0
5	1	1	0
6	1	1	0

(a) $ATE = -1$ ($SDO = -1$)
point A

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	0	1
2	0	0	1
3	0	0	1
4	1	0	0
5	1	0	0
6	1	0	0

(b) $ATE = \frac{1}{2}$ ($SDO = 0$)
point B

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	0	1
2	0	0	1
3	0	0	1
4	1	0	1
5	1	0	1
6	1	0	1

(c) $ATE = 1$ ($SDO = 1$)
point C

Figure 43.8: Examples of PO datasets. Exploring ATE extremes.

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	1	1
2	0	1	1
3	0	1	1
4	1	0	0
5	1	0	0
6	1	0	0

(a) $SDO = -1$ ($ATE = 0$)
point D

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	1	0
2	0	1	0
3	0	1	0
4	1	1	1
5	1	1	1
6	1	1	1

(b) $SDO = 0$ ($ATE = -\frac{1}{2}$)
point E

σ	\tilde{d}^σ	$y^\sigma(0)$	$y^\sigma(1)$
1	0	0	0
2	0	0	0
3	0	0	0
4	1	1	1
5	1	1	1
6	1	1	1

(c) $SDO = 1$ ($ATE = 0$)
point F

Figure 43.9: Examples of PO datasets. Exploring SDO extremes.

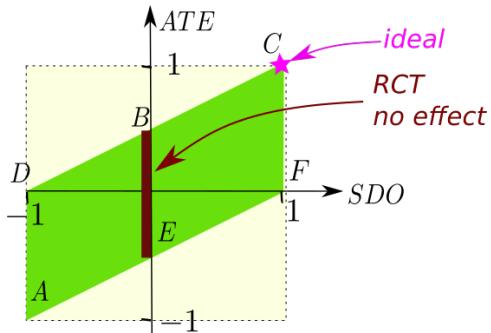


Figure 43.10: Green parallelogram is accessible region in (SDO, ATE) plane, assuming $y^\sigma \in \{0, 1\}$. Each of the six points A, B, ... F corresponds to one of the six tables in Figs. 43.8 and 43.9. Point C is ideal. The segment BE is the set of accessible points with zero effect in an RCT.

43.11 Matching Strata

For a situation described by the bnet G_{im+} , we can match *similar* individuals to fill the blank cells of Table 43.1. By “similar”, we mean that they have the same or almost the same value of x^σ .

IMPORTANT: Matching only makes sense if the individuals are **treatment blind** (i.e., have no knowledge of whether they are in the treated or control groups.)

43.11.1 Exact strata-match

For $\tilde{d} \in \{0, 1\}$ and all strata x , define the sets of individuals $A_{\tilde{d},x} = \{\sigma : \tilde{d}^\sigma = \tilde{d}, x^\sigma = x\}$, $A_x = A_{0,x} \cup A_{1,x}$ and $A = \cup_x A_x$. Let $N_{\tilde{d},x} = |A_{\tilde{d},x}|$, $N_x = |A_x|$ and $N = |A|$.

In an exact strata-match, we match each individual with $\tilde{d}^\sigma = 1, x^\sigma = x$ with exactly one individual with $\tilde{d}^\sigma = 0, x^\sigma = x$ and vice versa. Define a map $s : A \rightarrow A$ such that, for each x , $s(A_{0,x}) \subset A_{1,x}$ and $s(A_{1,x}) \subset A_{0,x}$. This assumes $A_{0,x}$ and $A_{1,x}$ are non-empty for all x . The purpose of map $s()$ is to fill in the missing data in the PO dataset. See Fig.43.3 for a pictorial representation of this.

	$y^\sigma(0)$	$y^\sigma(1)$
$\tilde{d}^\sigma = 0$	y^σ	$y^{s(\sigma)}$
$\tilde{d}^\sigma = 1$	$y^{s(\sigma)}$	y^σ

Table 43.3: Illustration of the purpose of the map $s()$. Note that $y^\sigma = y^\sigma(\tilde{d}^\sigma)$ and $y^{s(\sigma)} = y^\sigma(!\tilde{d}^\sigma)$, where $!0 = 1$ and $!1 = 0$.

Note that

$$E_{|x}[\tilde{d}^\sigma \underline{y}^\sigma] \neq E_{|x}[\tilde{d}^\sigma] E_{|x}[\underline{y}^\sigma] \quad (43.75)$$

but

$$E_{|x}[\underline{d}^\sigma \underline{y}^\sigma] = E_{|x}[\underline{d}^\sigma] E_{|x}[\underline{y}^\sigma] \quad (43.76)$$

because, by d-separation, at fixed x , \underline{y}^σ and \tilde{d}^σ are not independent but \underline{y}^σ and \underline{d}^σ are. Table 43.4 evaluates various expected values of the type $E_{|x}[\tilde{d}^\sigma \underline{y}^\sigma]$.

Recall that

$$ACE = SDO \quad (43.77a)$$

$$ATE = ATT P_{\underline{d}|x}(1|x) + ATU P_{\underline{d}|x}(0|x) \quad (43.77b)$$

$$ATT = \mathcal{Y}_{1|1,x} - \mathcal{Y}_{0|1,x} \quad (43.77c)$$

$$ATU = \mathcal{Y}_{1|0,x} - \mathcal{Y}_{0|0,x} \quad (43.77d)$$

	$y^\sigma(0)$	$y^\sigma(1)$
$\tilde{d}^\sigma = 0$	$E_{ x}[(1 - \tilde{d}^\sigma)\underline{y}^\sigma] = E[1 - \tilde{d}^\sigma]\mathcal{Y}_{0 0,x}$	$E_{ x}[(1 - \tilde{d}^\sigma)\underline{y}^{s(\sigma)}] = E[1 - \tilde{d}^\sigma]\mathcal{Y}_{1 0,x}$
	$E_{ x}\left[\frac{1}{N_{0,x}}\sum_{\sigma \in A_x}(1 - \tilde{d}^\sigma)\underline{y}^\sigma\right] = \mathcal{Y}_{0 0,x}$	$E_{ x}\left[\frac{1}{N_{0,x}}\sum_{\sigma \in A_x}(1 - \tilde{d}^\sigma)\underline{y}^{s(\sigma)}\right] = \mathcal{Y}_{1 0,x}$
$\tilde{d}^\sigma = 1$	$E_{ x}[\underline{d}^\sigma \underline{y}^\sigma] = E[\underline{d}^{s(\sigma)}]\mathcal{Y}_{0 1,x}$	$E_{ x}[\underline{d}^\sigma \underline{y}^\sigma] = E[\underline{d}^\sigma]\mathcal{Y}_{1 1,x}$
	$E_{ x}\left[\frac{1}{N_{1,x}}\sum_{\sigma \in A_x}\underline{d}^\sigma \underline{y}^{s(\sigma)}\right] = \mathcal{Y}_{0 1,x}$	$E_{ x}\left[\frac{1}{N_{1,x}}\sum_{\sigma \in A_x}\underline{d}^\sigma \underline{y}^\sigma\right] = \mathcal{Y}_{1 1,x}$

Table 43.4: Expected Values of the type $E_{|x}[\underline{d}^\sigma \underline{y}^\sigma]$.

$$SB = \mathcal{Y}_{0|1,x} - \mathcal{Y}_{0|0,x} \quad (43.77e)$$

$$SDO = \mathcal{Y}_{1|1,x} - \mathcal{Y}_{0|0,x} \quad (43.77f)$$

Eqs.(43.77) can be estimated from the data via the following estimators.

$$\widehat{ACE} = \widehat{SDO} \quad (43.78)$$

$$\widehat{ATE} = \frac{1}{N_x} [\widehat{ATT} N_{1,x} + \widehat{ATU} N_{0,x}] \quad (43.79)$$

$$= \frac{1}{N_x} \left[\sum_{\sigma \in A_x} \tilde{d}^\sigma [y^\sigma - y^{s(\sigma)}] + \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) [y^{s(\sigma)} - y^\sigma] \right] \quad (43.80)$$

$$= \frac{1}{N_x} \sum_{\sigma \in A_x} (2\tilde{d}^\sigma - 1) [y^\sigma - y^{s(\sigma)}] \quad (43.81)$$

$$\widehat{ATT} = \overbrace{\frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma y^\sigma}^{\mathcal{Y}_{1|1,x}} - \overbrace{\frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma y^{s(\sigma)}}^{\mathcal{Y}_{0|1,x}} \quad (43.82)$$

$$= \frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma [y^\sigma - y^{s(\sigma)}] \quad (43.83)$$

$$\widehat{ATU} = \overbrace{\frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) y^{s(\sigma)}}^{\mathcal{Y}_{1|0,x}} - \overbrace{\frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) y^\sigma}^{\mathcal{Y}_{0|0,x}} \quad (43.84)$$

$$= \frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) [y^{s(\sigma)} - y^\sigma] \quad (43.85)$$

$$\widehat{SB} = \overbrace{\frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma y^{s(\sigma)}}^{\mathcal{Y}_{0|1,x}} - \overbrace{\frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) y^\sigma}^{\mathcal{Y}_{0|0,x}} \quad (43.86)$$

$$\widehat{SDO} = \overbrace{\frac{1}{N_{1,x}} \sum_{\sigma \in A_x} \tilde{d}^\sigma y^\sigma}^{\mathcal{Y}_{1|1,x}} - \overbrace{\frac{1}{N_{0,x}} \sum_{\sigma \in A_x} (1 - \tilde{d}^\sigma) y^\sigma}^{\mathcal{Y}_{0|0,x}} \quad (43.87)$$

We've said before that strata matching makes no sense unless every individual is treatment blind. This means that if the individuals are not treatment blind, the above estimators for ATE , ATT , ATU , SB are invalid because they depend on $s()$. Only the estimator for SDO is valid because it doesn't depend on $s()$.

Let

$$g_d(x) = P_{d|x}(d|x) \quad (43.88)$$

for $d \in \{0, 1\}$. Note that $g_0(x) + g_1(x) = 1$. $g_1(x)$ is called the **propensity score**. One can do **propensity weighting** within the above estimators to improve their behavior. This is done by making the following replacements.

$$N_{1,x} \rightarrow N_x g_1(x^\sigma), \quad N_{0,x} \rightarrow N_x g_0(x^\sigma). \quad (43.89)$$

These replacements are equal under $\sum_{\sigma \in A_x}$ to the terms they replace.

Example, calculation of estimators for a treatment

For $\sigma \in \{1, 2, \dots, 10\}$, define

$$s(\sigma) = \begin{cases} \sigma + 5 & \text{if } \sigma \leq 5 \\ \sigma - 5 & \text{if } \sigma > 5 \end{cases} \quad (43.90)$$

Let $N(\mathcal{S})$ be the number of individuals σ that satisfy condition \mathcal{S} . For example, $N(\underline{\tilde{d}^\sigma} = \tilde{d})$ is the number of individuals such that $\underline{\tilde{d}^\sigma} = \tilde{d}$.

$$N_1 = N(\tilde{d}^\sigma = 1) = 5 \quad (43.91)$$

$$N_0 = N(\tilde{d}^\sigma = 0) = 5 \quad (43.92)$$

$$N = N_0 + N_1 = 10 \quad (43.93)$$

$$\mathcal{Y}_{1|1} = \frac{1}{N_1} \sum_{\sigma} \tilde{d}^\sigma y^\sigma = \frac{4}{5} \quad (43.94)$$

σ	\tilde{d}^σ	y^σ	$\tilde{d}^\sigma y^\sigma$	$(1 - \tilde{d}^\sigma)y^\sigma$	$\tilde{d}^\sigma y^{s(\sigma)}$	$(1 - \tilde{d}^\sigma)y^{s(\sigma)}$
1	0	0	0	0	0	0
2	0	0	0	0	0	1
3	0	1	0	1	0	1
4	0	1	0	1	0	1
5	0	1	0	1	0	1
6	1	0	0	0	0	0
7	1	1	1	0	0	0
8	1	1	1	0	1	0
9	1	1	1	0	1	0
10	1	1	1	0	1	0

Table 43.5: Estimators of treatment effects are calculated for this example.

$N(\tilde{d}, y)$	$y = 0$	$y = 1$
$\tilde{d} = 0$	2	3
$\tilde{d} = 1$	1	4

Table 43.6: $N(\underline{\tilde{d}^\sigma} = d, \underline{y^\sigma} = y)$ for the data in Table 43.5.

$$\mathcal{Y}_{0|0} = \frac{1}{N_0} \sum_{\sigma} (1 - \tilde{d}^\sigma) y^\sigma = \frac{3}{5} \quad (43.95)$$

$$\mathcal{Y}_{0|1} = \frac{1}{N_1} \sum_{\sigma} \tilde{d}^\sigma y^{s(\sigma)} = \frac{3}{5} \quad (43.96)$$

$$\mathcal{Y}_{1|0} = \frac{1}{N_0} \sum_{\sigma} (1 - \tilde{d}^\sigma) y^{s(\sigma)} = \frac{4}{5} \quad (43.97)$$

$$ATT = \mathcal{Y}_{1|1} - \mathcal{Y}_{0|1} = \frac{1}{5} \quad (43.98)$$

$$ATE = ATT = ATU = SDO = \frac{1}{5}, \quad SB = 0 \quad (43.99)$$

This example is unusual in that it has a single stratum x , and for that stratum, the treated and untreated populations are **balanced** (of equal size). Also, the map $s()$ is 1-1 onto. If, for instance, $s(\sigma) = 6$ for all $\sigma \in A_0$ and $s(\sigma) = 5$ for $\sigma \in A_1$, then ATE, ATT, ATU, SDO would not all be same, and SB would not be zero. In fact,

whenever there is a single balanced stratum and the map $s()$ is 1-1 onto, Eq.43.99 can be proven to be true using the methods of section 43.9.

43.11.2 Approximate strata-match

It is very often the case that one can't find for a given individual σ another individual that has opposite d^σ but exactly the same value of x^σ . In such cases, one can discard all matchless individuals. But that would entail a loss of precious information. Instead of discarding orphans, a better way is to relax our demands and match individual σ with another individual $s(\sigma)$ such that x^σ and $x^{s(\sigma)}$ are very close in some metric. Alternatively, the matching individual might not be real; it might be a composite of individuals.

More precisely, for some arbitrary parameter $\epsilon > 0$, and an individual σ with $d^\sigma = 1$, define the **strata-matching set** $\mathcal{M}_\epsilon(\sigma)$ by⁴

$$\mathcal{M}_\epsilon(\sigma) = \{s : d^\sigma = 1, d^s = 0, \text{dist}(x^\sigma, x^s) \leq \epsilon\}, \quad (43.100)$$

where

$$\text{dist}(x^\sigma, x^s) = [x^\sigma]^T [\Sigma]^{-1} x^s, \quad (43.101)$$

where $\Sigma = \langle x^\sigma, [x^s]^T \rangle$. This metric $\text{dist}(x^\sigma, x^s)$ is called the **Mahalanobis distance**. We will call the case $\epsilon = 0$ an **exact strata-match**, and the case $\epsilon \neq 0$ an **approximate strata-match**. To do an approximate strata-match, replace $y^{s(\sigma)}$ by $\langle y \rangle^\sigma$ in the estimators given above for an exact strata-match. $\langle y \rangle^\sigma$ is defined by

$$\langle y \rangle^\sigma = \frac{1}{|\mathcal{M}_\epsilon(\sigma)|} \sum_{s \in \mathcal{M}_\epsilon(\sigma)} y^s. \quad (43.102)$$

Ref.[3] calculates the mean and variance of estimator \widehat{ATT} . The mean is biased, but one can define a new bias-corrected estimator.

43.11.3 Positivity

Positivity is defined as the requirement that for all layers x ,

$$0 < P(d^\sigma = 1 | x^\sigma = x) < 1 \quad (43.103)$$

or, equivalently,

$$P(d^\sigma = 1 | x^\sigma = x) > 0 \quad \text{and} \quad P(d^\sigma = 0 | x^\sigma = x) > 0. \quad (43.104)$$

⁴ One can use an ϵ that depends on σ . Let $\epsilon(\sigma, 5)$ be the radius necessary so that $\mathcal{M}_{\epsilon(\sigma, 5)}(\sigma)$ contains exactly 5 elements s . Thus, $\mathcal{M}_{\epsilon(\sigma, 5)}(\sigma)$ contains the s of the 5 points x^s that are the nearest neighbors of x^σ in the $\text{dist}()$ metric.

In other words, for each layer x , there is a non-zero probability of being both treated and untreated.

Recall that

$$\mathcal{Y}_{d|\tilde{d},x} = \sum_y y P_{\underline{y}(d)|\tilde{d},x}(y|\tilde{d},x) . \quad (43.105)$$

Also recall that the estimator Eq.(43.83) for $ATT|_x$ divides by $N_{1,x} = P(d = 1|x)N_x$ and the estimator Eq.(43.85) for $ATU|_x$ divides by $N_{0,x} = P(d = 0|x)N_x$. The estimator Eq.43.81 for $ATE|_x$ divides by neither $N_{0,x}$ nor $N_{1,x}$ so it is safe.

If positivity is violated, then for some layer x , $\mathcal{Y}_{d|\tilde{d}=0,x}$ or $\mathcal{Y}_{d|\tilde{d}=1,x}$ is undefined. Furthermore, the estimator $ATT|_x$ or $ATU|_x$ is undefined. If $ATT|_x$ (or any other treatment effect) can be estimated, one says it is **identifiable** (i.e., calculable). If Positivity is violated, then either $ATT|_x$ or $ATU|_x$ is not identifiable.

When $P(d^\sigma|x^\sigma = x)$ becomes 0 or 1 for some x , the arrow $\underline{x} \rightarrow \underline{d}$ becomes deterministic for some x . This situation is the very antithesis of RCTs, wherein the influence exerted by \underline{x}^σ on \underline{d}^σ is uniformly random and therefore ignorable. Hence, it is perhaps not too surprising that a violation of positivity makes ATT or ATU non-identifiable.

43.12 Propensity Score

It is often the case that the discrete vector \underline{x}^σ has too many possible values to make matching possible. In such cases, it is convenient to map the space of vectors \underline{x}^σ to the real line. One very convenient choice for that map is the **propensity score**, which is defined as

$$g(x^\sigma) = P(d^\sigma = 1|x^\sigma) . \quad (43.106)$$

The propensity score is usually approximated by a sigmoid function using logistic regression⁵

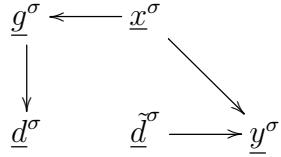
$$g(x^\sigma) = \text{smoid}(\alpha + \beta x^\sigma) \quad (43.107)$$

To use the propensity score, one replaces the bnet G_{im+} by the bnet G_{ps} shown in Fig.43.11. The TPMs, printed in blue, for the 2 nodes of G_{ps} that differ from the nodes of G_{im+} , are as follows:

$$P(g^\sigma|x^\sigma) = \delta(g^\sigma, g(x^\sigma)) \quad (43.108)$$

$$P(d^\sigma|g^\sigma) = g^\sigma d^\sigma + (1 - g^\sigma)(1 - d^\sigma) \quad (43.109)$$

⁵ The sigmoid function is defined in Chapter Notational Conventions and Preliminaries to be $\text{smoid}(x) = 1/(1 - e^{-x})$.



G_{ps}

Figure 43.11: Bnet G_{ps} used when doing propensity scoring.

Note that these TPMs are self-consistent because

$$P(d|x) = \sum_g P(d|g)P(g|x) \quad (43.110)$$

$$= g(x)d + [1 - g(x)](1 - d) \quad (43.111)$$

$$= P(d=1|x)d + [1 - P(d=1|x)](1 - d) \quad (43.112)$$

$$= P(d|x) \quad (43.113)$$

We would like to do **propensity score strata-matching** by matching g-strata instead of x-strata. PO calculations for x-strata matching use the TPMs for $P(d|x)$, $P(x)$ and $P(y|d, x)$. To do g-strata matching using the same equations, but with x replaced by g , we would need to solve for $P(d|g)$, $P(g)$ and $P(y|d, g)$ in terms of $P(d|x)$, $P(x)$ and $P(y|d, x)$. We solve for those next.

From the TPMs for G_{ps} , one has

$$\boxed{P(d|g) = gd + (1 - g)(1 - d)} \quad (43.114)$$

and

$$\boxed{P(g) = \sum_x \overbrace{\delta(g, g(x))}^{P(g|x)} P(x)} . \quad (43.115)$$

Next, note that

$$P(y|d, g) = \sum_x P(y|d, x)P(x|g) \quad (43.116)$$

so we need to find $P(x|g)$. Since

$$P(x|g) = \frac{P(g|x)P(x)}{P(g)} \quad (43.117)$$

$$= \frac{\delta(g, g(x))P(x)}{P(g)} \quad (43.118)$$

we finally get

$$\boxed{P(y|d, g) = \sum_x P(y|d, x) \frac{\delta(g, g(x))P(x)}{P(g)}}. \quad (43.119)$$

43.13 Multi-time PO bnets (Panel Data)

In this section, we will discuss Multi-time PO bnets (MT-PO).

A **time-series** is a function $f : D \rightarrow \mathbb{R}$ whose domain D is a discrete set. A time-series usually describes a single unit σ (i.e., an individual) in a population.

An **observational study (or analysis or model)** can be cross-sectional or longitudinal. A **cross-sectional study** collects and analyzes a **cross-sectional dataset**; i.e., a dataset for a population at a single time. A **longitudinal study or panel study** collects and analyzes a **longitudinal dataset**; i.e., a dataset for a population at multiple times. Thus, a longitudinal study consists of one or more time-series.

Let $\mathcal{T} = \{t_0, t_1, \dots, t_{ntimes-1}\}$. For any time-series $a_t : \mathcal{T} \rightarrow \mathbb{R}$, define

$$E_t a_t = \frac{1}{ntimes} \sum_{t \in \mathcal{T}} a_t \quad (43.120)$$

$$\Delta_t a_t = a_t - E_t a_t \quad (43.121)$$

$$\langle a_t, b_t \rangle_t = E_t \Delta_t a_t \Delta_t b_t \quad (43.122)$$

Consider a quantity a_t^σ that is a function of the time t and of the particular unit σ in a population. a_t^σ is said to be a **fixed (in time) effect** if it is t -independent. a_t^σ is said to be a **homogeneous effect** (antonym: **heterogeneous effect**) if it is σ -independent. Henceforth, we will avoid using the word “effect” for these, because that word has already been used for something else in PO theory. Instead, we will use the word “quantity”.

Fig.43.12 gives an example of a multi-time PO bnet (MT-PO). Note that in this example, \underline{x}^σ and \underline{u}^σ are fixed quantities (i.e., they are t -independent). \underline{u}^σ is an unobserved confounder and \underline{x}^σ is an observed confounder. For convenience and

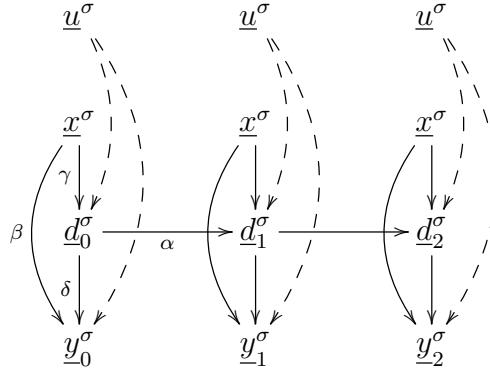


Figure 43.12: Example of multi-time PO bnet with fixed quantities $\underline{x}^\sigma, \underline{u}^\sigma$. The 3 nodes \underline{x}^σ should be identified as a single node. Likewise, the 3 nodes \underline{u}^σ should be identified as a single node.

simplicity, we will assume linear deterministic TPMs for the internal (i.e., non-root) nodes. The TPMs for the bnet Fig.43.12, printed in blue, are as follows:

$$P(x^\sigma) = P_{\underline{x}}(x^\sigma) \quad (43.123)$$

$$P(u^\sigma) = P_{\underline{u}}(u^\sigma) \quad (43.124)$$

$$P(y_t^\sigma | d_t^\sigma, x^\sigma, u^\sigma) = \mathbb{1}(y_t^\sigma = \delta d_t^\sigma + \beta x^\sigma + u^\sigma) \quad (43.125)$$

$$P(d_{t+1}^\sigma | d_t^\sigma, x^\sigma, u^\sigma) = \mathbb{1}(d_{t+1}^\sigma = \alpha d_t^\sigma + \gamma x^\sigma + u^\sigma) \quad (43.126)$$

Taking time averages of the treatment dose and treatment outcome, we get

$$E_t \underline{y}_t^\sigma = \delta E_t \underline{d}_t^\sigma + \beta \underline{x}^\sigma + \underline{u}^\sigma , \quad (43.127)$$

$$E_t \underline{d}_{t+1}^\sigma = \alpha E_t \underline{d}_t^\sigma + \gamma \underline{x}^\sigma + \underline{u}^\sigma . \quad (43.128)$$

Subtracting the time averages from the quantities being averaged, we get

$$\Delta_t \underline{y}_t^\sigma = \delta \Delta_t \underline{d}_t^\sigma , \quad (43.129)$$

$$\Delta_t \underline{d}_{t+1}^\sigma = \alpha \Delta_t \underline{d}_t^\sigma . \quad (43.130)$$

This allows us to find estimators for δ and α :

$$E_\sigma \left\langle \underline{y}_t^\sigma, \underline{y}_t^\sigma \right\rangle_t = \delta E_\sigma \left\langle \underline{y}_t^\sigma, \underline{d}_t^\sigma \right\rangle_t \quad (43.131)$$

$$\delta = \frac{E_\sigma \left\langle \underline{y}_t^\sigma, \underline{y}_t^\sigma \right\rangle_t}{E_\sigma \left\langle \underline{y}_t^\sigma, \underline{d}_t^\sigma \right\rangle_t} \quad (43.132)$$

$$E_\sigma \left\langle \underline{d}_{t+1}^\sigma, \underline{d}_{t+1}^\sigma \right\rangle_t = \alpha E_\sigma \left\langle \underline{d}_{t+1}^\sigma, \underline{d}_t^\sigma \right\rangle_t \quad (43.133)$$

$$\alpha = \frac{E_\sigma \left\langle \underline{d}_{t+1}^\sigma, \underline{d}_{t+1}^\sigma \right\rangle_t}{E_\sigma \left\langle \underline{d}_{t+1}^\sigma, \underline{d}_t^\sigma \right\rangle_t} \quad (43.134)$$

As shown in Fig.43.13, subtraction of time averages from each node removes the confounder nodes from the bnet of Fig.43.12 (However, this assumes that the confounders are time independent and that the TPMs for the internal nodes are linear deterministic, two very strong assumptions).

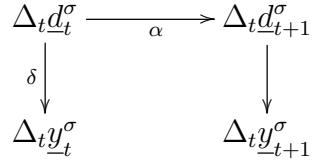


Figure 43.13: time-average-subtracted (TAS) bnet for the bnet of Fig.43.12.

Chapter 44

Program evaluation and review technique (PERT)

This chapter is based on Refs.[39] and [84].

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.44.2 for an example of a PERT diagram). The nodes \underline{E}_i for $i = 1, 2, \dots, ne$ 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 $i \rightarrow j$

$$D_{i \rightarrow j} = \frac{1}{6}(DO_{i \rightarrow j} + DP_{i \rightarrow j} + 4DM_{i \rightarrow j}) \quad (44.1)$$

Duration Variance of activity $i \rightarrow j$

$$V_{i \rightarrow j} = \left(\frac{DO_{i \rightarrow j} - DP_{i \rightarrow j}}{DM_{i \rightarrow j}} \right)^2 \quad (44.2)$$

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
 $TEF_{i \rightarrow j} = TES_i + D_{i \rightarrow j}$ = Earliest finish time for activity $i \rightarrow j$.
 $TLF_{i \rightarrow j} = TLS_j - D_{i \rightarrow j}$ = Latest finish time for activity $i \rightarrow j$. 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 $D_{i \rightarrow j}$ 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

$$TES_i = \max_{a \in pa(i)} (\underbrace{TES_a + D_{a \rightarrow i}}_{TEF_{a \rightarrow i}}) \quad (44.3)$$

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

$$TLS_i = \min_{b \in ch(i)} (\underbrace{TLS_b - D_{i \rightarrow b}}_{TLF_{i \rightarrow b}}) \quad (44.4)$$

¹ In the popular educational literature, the edge variables $TEF_{i \rightarrow j}$ and $TLF_{i \rightarrow j}$ 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 $TEF_{i \rightarrow j}$ and $TLF_{i \rightarrow j}$ in a node makes some sense but not for DAGs. You will notice that giving specific names to the variables $TEF_{i \rightarrow j}$ and $TLF_{i \rightarrow j}$ 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.

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.(44.3) and (44.4) are illustrated in Fig.44.1.

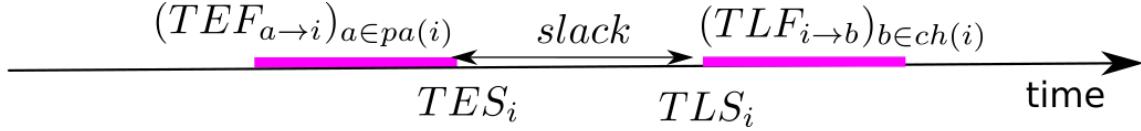


Figure 44.1: TES_i defined from info received from parents of i and TLS_i defined from info received from children of i .

44.1 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.[39], except for part (e) about bnets, which is our own.

Question: For the PERT diagram of Fig.44.2, 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.44.3.

Answer to (b) The critical path is given in red in Fig.44.3. 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.

Answer to (c) The mean and variance of the total duration are calculated in Table 44.1.

Answer to (d)

$$P(\underline{x} < 225) = P\left[\frac{\underline{x} - \mu}{\sigma} \leq \frac{225 - 220}{\sqrt{7.73}}\right] \quad (44.5)$$

$$= P[\underline{z} \leq 1.80] \quad (44.6)$$

$$= 0.9641 \quad (44.7)$$

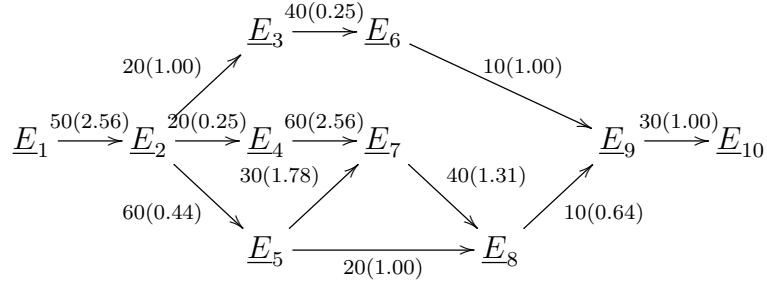


Figure 44.2: Example of a PERT diagram. The numbers attached to the arrows are the duration times $D_{i \rightarrow j}$ in days followed by, enclosed in parentheses, the variance $V_{i \rightarrow j}$ of that duration. The info given in this PERT diagram was derived from a DT table in Ref.[39]. The info 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.44.3.

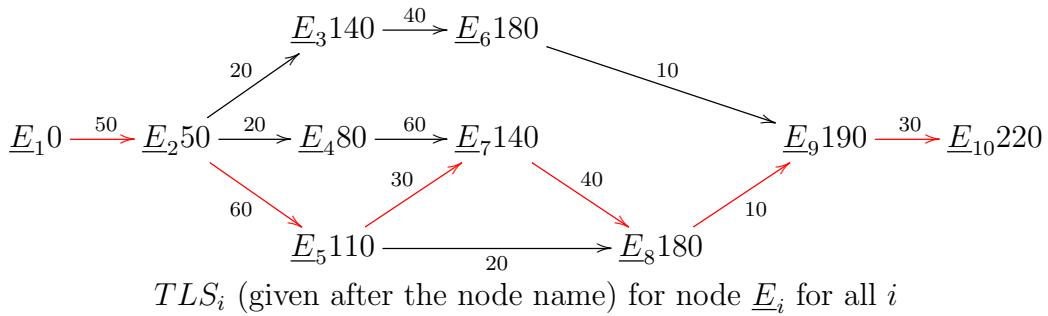
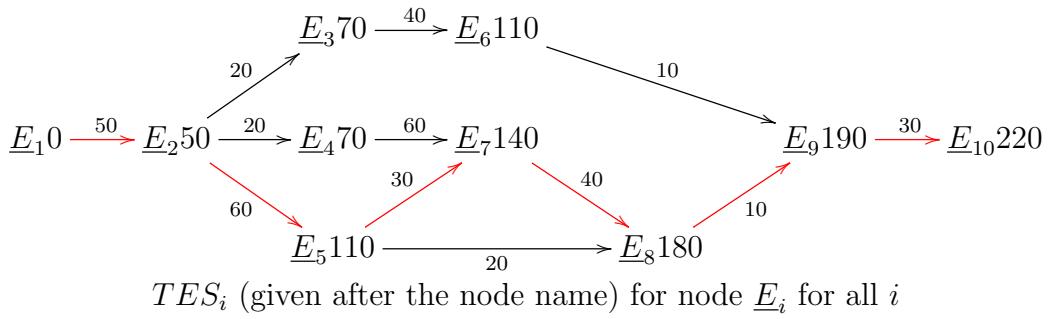


Figure 44.3: Results of calculating TES_i for all i via a forward pass, followed by calculating TLS_i for all i via a backward pass. Critical path indicated in red.

Answer to (e) Define 2 bnets.

1. The first PERT bnet is for calculating TES_i for all i and is given by Fig.44.4.

edge $i \rightarrow j$	duration $D_{i \rightarrow j}$	variance $V_{i \rightarrow j}$
A (1 → 2)	50	2.56
D (2 → 5)	60	0.44
G (5 → 7)	30	1.78
J (7 → 8)	40	1.31
K (8 → 9)	10	0.64
L (9 → 10)	30	1.00
Total	220	7.73

Table 44.1: Calculation of mean and variance of total duration along critical path.

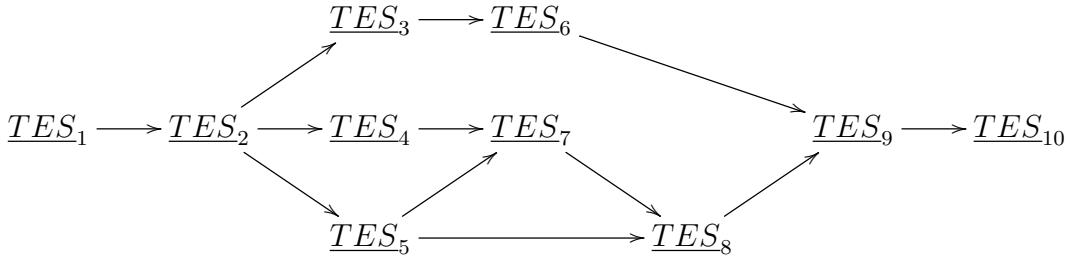


Figure 44.4: bnet for TES_i calculation.

The node TPMs, printed in blue, for the bnet Fig.44.4 are given by (this equation is to be evaluated recursively by a forward pass through the bnet):

$$P(TES_i | (TES_a)_{a \in pa(i)}) = \delta(TES_i, \max_{a \in pa(i)} (TES_a + D_{a \rightarrow i})) \quad (44.8)$$

2. The second PERT bnet is for calculating TLS_i for all i and is given by Fig.44.5. Note that the directions of all the arrows in the PERT diagram Fig.44.2 have been reversed so Fig.44.5 is a time reversed graph.

The node TPMs, printed in blue, for the bnet Fig.44.5 are given by (this equation is to be evaluate recursively by a backward pass through the bnet):

$$P(TLS_i | (TLS_b)_{b \in pa(i)}) = \delta(TLS_i, \min_{b \in pa(i)} (TLS_b - D_{b \rightarrow i}^T)) , \quad (44.9)$$

where $D_{i \rightarrow j}^T = D_{j \rightarrow i}$.

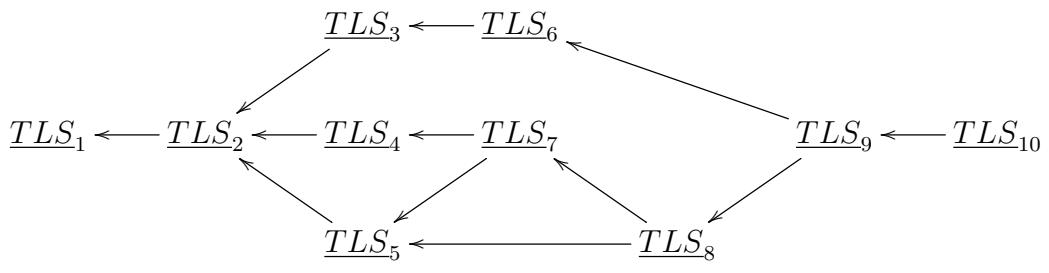


Figure 44.5: bnet for TLS_i calculation.

Chapter 45

Random Forest and Bagging

This chapter is based on Refs.[50] and [85].

Chapter 12 defines decision trees (dtrees) and explains how to construct them. A Random Forest (RF) is an ensemble of dtrees. The RF algorithm is a method of, given a dataset, constructing a RF and averaging over the classifier functions of the RF to produce an ensemble classifier.

The RF algo uses the method of Boostrap Aggregating (aka Bagging), which is discussed in detail below. Bagging is a method of constructing an ensemble of datasets (called **bootstrap datasets or bags**) that are fairly uncorrelated. Each of these bags is used to train a **bag-classifier**. The bag-classifiers are averaged over to produce an **ensemble classifier, or e-classifier** for short. Bagging can be used to train any type of bag-classifier, but it was invented with dtrees in mind, and is still most commonly used to train dtrees.

Chapter 1 is on AdaBoosting, which is another method, besides Bagging, of constructing a classifier function from an ensemble of classifier functions. These two methods are most commonly applied to dtrees: AdaBoosting for an ensemble of tree stumps, and Bagging for a RF (which is an ensemble of dtrees that are usually much more complicated than tree stumps). Both methods are highly effective in mitigating overfitting, a common problem with simple dtrees. Overfitting is characterized by low bias/high variance. Bagging decreases variance without significantly affecting bias.

45.1 Bagging (with fully-featured bags)

In this section we discuss the bagging algorithm. As already mentioned, bagging is usually used to train dtrees. In this section, we explain bagging in general, not just for dtrees.

Let $L = [0, 1, 2, \dots, nsam - 1]$ be a list of individuals (samples) in a population. In this chapter, we will use the notation $A^\sigma = A[\sigma]$ and $\vec{A} = [A^\sigma : \sigma \in L]$ for a list (vector, 1-D array) indexed by L . We will refer to $DS = (\vec{x}, \vec{y})$ where $x^\sigma \in S_x$, $y^\sigma \in S_y$, as a dataset. If L_j is a list (possibly with duplicate items) such that

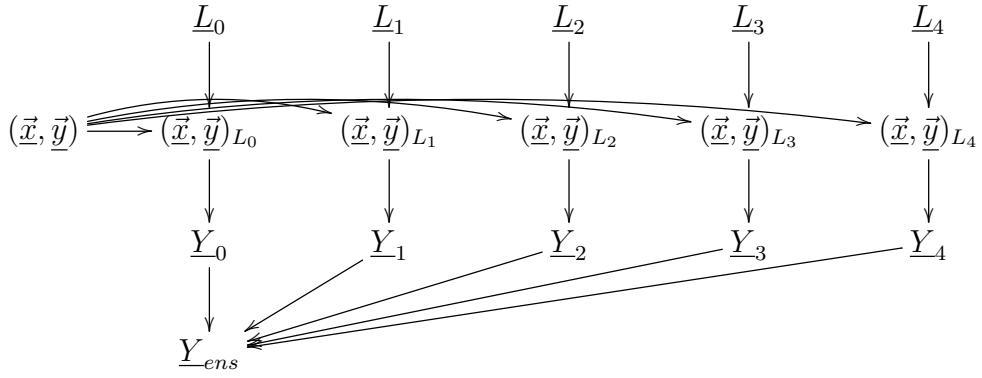


Figure 45.1: Bnet for Random Forest (RF) with 5 bags.

$\text{set}(L_j) \subset \text{set}(L)$, then define $DS_j = (\vec{x}, \vec{y})_{L_j} = ((x^\sigma)_{\sigma \in L_j}, (y^\sigma)_{\sigma \in L_j})$. We will refer to DS_j as the **restriction of (\vec{x}, \vec{y}) to L_j** .

We will refer to a function $Y : S_{\underline{x}} \rightarrow S_{\underline{c}}$ as a classifier. It maps vector of vector of features $x \in S_{\underline{x}}$ to a class $c \in S_{\underline{c}}$. Below, Y_b for all b and Y_{ens} are classifiers.

Fig.45.1 is a bnet that encapsulates the RF algo. The TMPs, printed in blue, for the nodes of bnet Ref.45.1, are as follows.

Let $b \in \{0, 1, 2, \dots, nbags - 1\} = B$ and $\sigma \in L$. Let $L_b^\sigma \in L$ and

$$P(L_b^\sigma) = 1/nsam \quad (45.1)$$

In other words, each item in list L_b is chosen from the items of list L , uniformly at random with replacements. $|L_b| = |L|$ (same size as original). L_b can have duplicate items and be missing items from L .

$$P((\vec{x}, \vec{y})_{L_b} | (\vec{x}, \vec{y}), L_b) = \mathbb{1}(\ (\vec{x}, \vec{y})_{L_b} = \text{restriction of } (\vec{x}, \vec{y}) \text{ to } L_b \) \quad (45.2)$$

We will refer to (\vec{x}, \vec{y}) as the **original dataset** and to the $(\vec{x}, \vec{y})_{L_b}$ for $b \in B$ as the **bootstrap datasets or bags**.

$$P(Y_b | (\vec{x}, \vec{y})_{L_b}) = \mathbb{1}(\ Y_b(\cdot) = \text{classifier trained on dataset } (\vec{x}, \vec{y})_{L_b}. \) \quad (45.3)$$

$$P(Y_{ens} | (Y_b)_{b \in B}) = \prod_{\sigma} \mathbb{1}(\ Y_{ens}(x^\sigma) = \text{majority}(\{Y_b(x^\sigma) : b \in B\}) \) \quad (45.4)$$

The **majority()** function can be replaced by an average $\frac{1}{nbags} \sum_b$ in case the set of classes $S_{\underline{c}}$ equals \mathbb{R} rather than a finite set. We will refer to Y_{ens} as the **ensemble classifier (e-classifier)** and to the Y_b as the **bag-classifiers**.

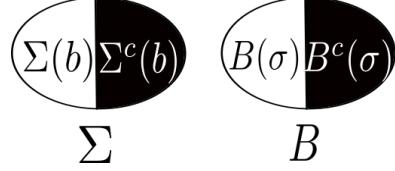


Figure 45.2: $\Sigma(b)$ and $\Sigma^c(b)$ are disjoint sets whose union is Σ . $B(\sigma)$ and $B^c(\sigma)$ are disjoint sets whose union is B .

Define (these definitions are illustrated in Fig.45.2)

$$\Sigma = \text{set}(L), \quad \Sigma(b) = \text{set}(L_b), \quad \Sigma^c(b) = \Sigma - \Sigma(b) \quad (45.5)$$

and

$$B(\sigma) = \{b \in B : \sigma \in \Sigma(b)\}, \quad B^c(\sigma) = B - B(\sigma) \quad (45.6)$$

$\Sigma(b)$ is the set of **in-the-b-bag individuals** and $\Sigma^c(b)$ is the set of **out-of-the-b-bag (OOB) individuals**. $B(\sigma)$ is the set of bags that contain individual σ , and $B^c(\sigma)$ is the set of bags that don't.

The **OOB error** is defined as

$$err = \sum_{\sigma \in L} \mathbb{1}(B^c(\sigma) \neq \emptyset) \mathbb{1}(y^\sigma \neq \text{majority}([Y_b(x^\sigma) : b \in B^c(\sigma)])) . \quad (45.7)$$

Empirical results supposedly show that OOB error is comparable in accuracy to the error calculated by doing cross-validation (CV) (see Chapter 11), although CV error is considered more dependable.

45.2 Bagging (with randomly-shortened bags)

Suppose the feature vector x^σ in the dataset $DS = (\vec{x}, \vec{y})$ has nf components; i.e., $x^\sigma = (x_0^\sigma, x_1^\sigma, \dots, x_{nf-1}^\sigma) \in S_{\underline{x}_0} \times S_{\underline{x}_1} \times \dots \times S_{\underline{x}_{nf-1}} = S_{\underline{x}}$.

For each bag DS_b , one chooses at random $nf' = \sqrt{nf}$ out of the nf features, and discards the remaining features from DS_b , thus producing a new, **randomly-shortened-bag (rs-bag)** DS'_b . Each rs-bag is then used to train a bag-classifier, usually a dtree, using the methods for dtree SL described in Chapter 12. Using rs-bags is called the **random subspace method**. The reason for using rs-bags is that they further decorrelate the set of bags used to train bag-classifiers.

Chapter 46

Recurrent Neural Networks

This chapter is mostly based on Ref.[20].

This chapter assumes you are familiar with the material and notation of Chapter 39 on plain Neural Nets.

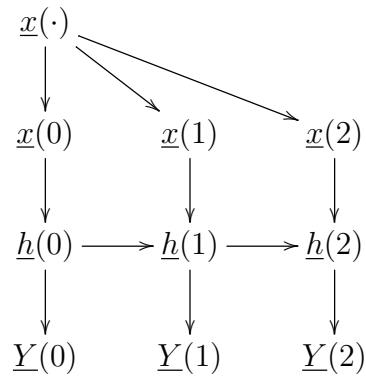


Figure 46.1: Simple example of RNN with $T = 3$

Suppose

T is a positive integer.

$t = 0, 1, \dots, T - 1$,

$\underline{x}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, nx - 1$,

$\underline{h}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, nh - 1$,

$\underline{Y}_i(t) \in \mathbb{R}$ for $i = 0, 1, \dots, ny - 1$,

$W^{h|x} \in \mathbb{R}^{nh \times nx}$,

$W^{h|h} \in \mathbb{R}^{nh \times nh}$,

$W^{y|h} \in \mathbb{R}^{ny \times nh}$,

$b^y \in \mathbb{R}^{ny}$,

$b^h \in \mathbb{R}^{nh}$.

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.46.1 with arbitrary T . The node TPMs, printed in blue, for this bnet, are as follows.

$$P(x(\cdot)) = \text{given} \quad (46.1)$$

$$P(x(t)) = \delta(x(t), [x(\cdot)]_t) \quad (46.2)$$

$$P(h(t) | h(t-1), x(t)) = \delta(h(t), \mathcal{A}(W^{h|x}x(t) + W^{h|h}h(t-1) + b^h)) , \quad (46.3)$$

where $h(-1) = 0$.

$$P(Y(t) | h(t)) = \delta(Y(t), \mathcal{A}(W^{y|h}h(t) + b^y)) \quad (46.4)$$

Define

$$W^h = [W^{h|x}, W^{h|h}, b^h] , \quad (46.5)$$

and

$$W^y = [W^{y|h}, b^y] . \quad (46.6)$$

The bnet of Fig.46.1 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.46.2.

Let $\sigma = 0, 1, \dots, nsam(\vec{x}) - 1$ be the labels for a minibatch of samples. Below, we will write $A^{[\sigma]} = A[\sigma] = [\vec{A}]^\sigma$ for the σ component of any vector \vec{A} . The node TPMs, printed in blue, for bnet Fig.46.2, are as follows.

$$P(x(\cdot)^{[\sigma]}) = \text{given} \quad (46.7)$$

$$P(x(t)^{[\sigma]}) = \delta(x(t)^{[\sigma]}, [x(\cdot)]_t^{[\sigma]}) \quad (46.8)$$

$$P(h(t)^{[\sigma]} | h(t-1)^{[\sigma]}, x(t)^{[\sigma]}) = \delta(h(t)^{[\sigma]}, \mathcal{A}(W^{h|x}x(t)^{[\sigma]} + W^{h|h}h(t-1)^{[\sigma]} + b^h)) \quad (46.9)$$

$$P(Y(t)^{[\sigma]} | h(t-1)^{[\sigma]}) = \delta(Y(t)^{[\sigma]}, \mathcal{A}(W^{y|h}h(t-1)^{[\sigma]} + b^y)) \quad (46.10)$$

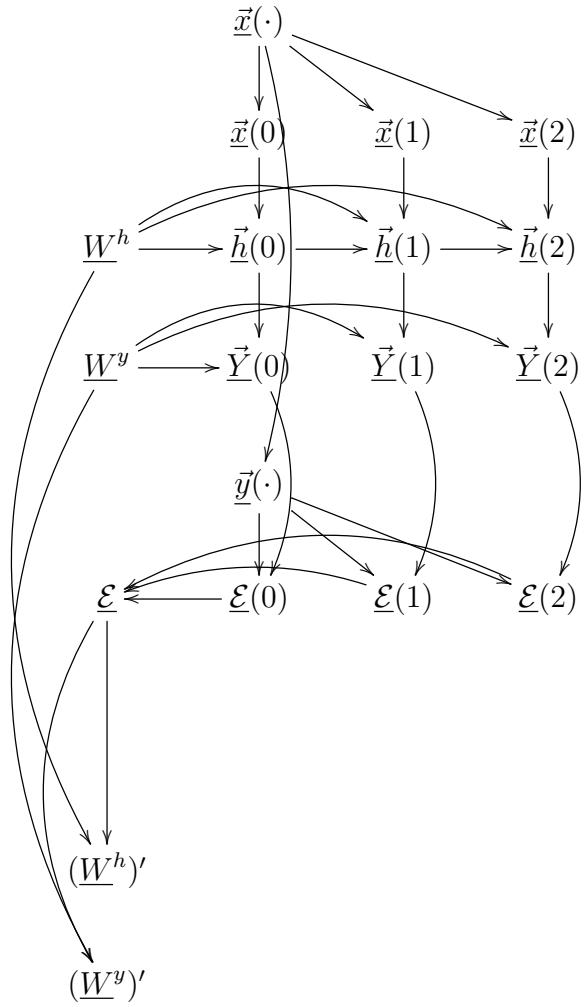


Figure 46.2: RNN bnet used to optimize parameters W^h and W^y of RNN bnet Fig.46.1.

$$P(y(\cdot)^{[\sigma]} \mid x(\cdot)^{[\sigma]}) = \text{given} \quad (46.11)$$

$$P(\mathcal{E}(t) \mid \vec{y}(t), \vec{Y}(t)) = \frac{1}{nsam(\vec{x})} \sum_{\sigma} d(y(t)^{[\sigma]}, Y(t)^{[\sigma]}) , \quad (46.12)$$

where

$$d(y, Y) = |y - Y|^2 . \quad (46.13)$$

If $y, Y \in [0, 1]$, one can use this instead

$$d(y, Y) = XE(y \rightarrow Y) = -y \ln Y - (1 - y) \ln(1 - Y). \quad (46.14)$$

$$P(\mathcal{E} | \mathcal{E}(\cdot)) = \delta(\mathcal{E}, \sum_t \mathcal{E}(t)) \quad (46.15)$$

For $a = h, y$,

$$P(W^a) = \text{given}. \quad (46.16)$$

The first time it is used, W^a is arbitrary. Afterwards, it is determined by previous horizontal stage.

$$P((W^a)' | \mathcal{E}, W^a) = \delta((W^a)', W^a - \eta^a \partial_{W^a} \mathcal{E}). \quad (46.17)$$

$\eta^a > 0$ is the learning rate for W^a .

46.1 Language Sequence Modeling

Estimate $P(x(\cdot))$ empirically. 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 $x(2) \sim P(x(2)|x(0), x(1))$.

46.2 Other types of RNN

Let $\mathcal{T} = \{0, 1, \dots, T - 1\}$, 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 (e.g., good choice of initial W^h and

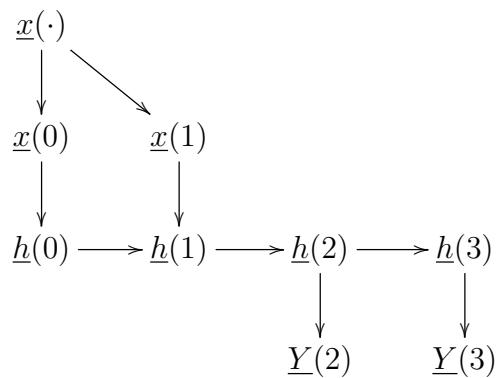


Figure 46.3: RNN bnet of the many to many kind. This one can be used for translation. $x(0)$ and $x(1)$ might denote two words of an English sentence, and $Y(2)$ and $Y(3)$ might be their Italian translation.

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).

46.2.1 Long Short Term Memory (LSTM) unit (1997)

This section is based on Wikipedia article Ref.[73]. In this section, \odot will denote the Hadamard matrix product (elementwise product).

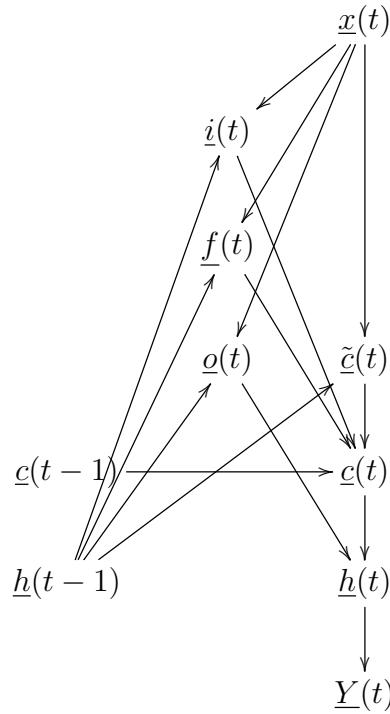


Figure 46.4: bnet for a Long Short Term Memory (LSTM) unit.

Let

$\underline{x}(t) \in \mathbb{R}^{nx}$: **input state vector** to the LSTM unit

$\underline{f}(t) \in \mathbb{R}^{nh}$: **forget activation vector**

$\underline{i}(t) \in \mathbb{R}^{nh}$: **input activation vector**

$\underline{o}(t) \in \mathbb{R}^{nh}$: **output activation vector**

$\underline{h}(t) \in \mathbb{R}^{nh}$: **hidden state vector**

$\underline{\tilde{c}}(t) \in \mathbb{R}^{nh}$: **cell activation vector**

$\underline{c}(t) \in \mathbb{R}^{nh}$: **cell state vector**

$\underline{Y}(t) \in \mathbb{R}^{ny}$: **classification** of $x(t)$.

$W \in \mathbb{R}^{nh \times nx}$, $U \in \mathbb{R}^{nh \times nh}$ and $b \in \mathbb{R}^{nh}$: weight matrices and bias vectors, parameters learned by training.

$\mathcal{W}^{y|h} \in \mathbb{R}^{ny \times nh}$: weight matrix

Fig.46.4 is a bnet for a LSTM unit. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(f(t)|x(t), h(t-1)) = \mathbb{1}(\quad f(t) = \text{smoid}(W^{f|x}x(t) + U^{f|h}h(t-1) + b^f) \quad) , \quad (46.18)$$

where $h(-1) = 0$.

$$P(i(t)|x(t), h(t-1)) = \mathbb{1}(\quad i(t) = \text{smoid}(W^{i|x}x(t) + U^{i|h}h(t-1) + b^i) \quad) \quad (46.19)$$

$$P(o(t)|x(t), h(t-1)) = \mathbb{1}(\quad o(t) = \text{smoid}(W^{o|x}x(t) + U^{o|h}h(t-1) + b^o) \quad) \quad (46.20)$$

$$P(\tilde{c}(t)|x(t), h(t-1)) = \mathbb{1}(\quad \tilde{c}(t) = \tanh(W^{c|x}x(t) + U^{c|h}h(t-1) + b^c) \quad) \quad (46.21)$$

$$P(c(t)|f(t), c(t-1), i(t), \tilde{c}(t)) = \mathbb{1}(\quad c(t) = f(t) \odot c(t-1) + i(t) \odot \tilde{c}(t) \quad) \quad (46.22)$$

$$P(h(t)|o(t), c(t)) = \mathbb{1}(\quad h(t) = o(t) \odot \tanh(c(t)) \quad) \quad (46.23)$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h}h(t) + b^y) \quad) \quad (46.24)$$

46.2.2 Gated Recurrence Unit (GRU) (2014)

This section is based on Wikipedia article Ref.[59]. In this section, \odot will denote the Hadamard matrix product (elementwise product).

GRU is a more recent (17 years later) attempt at simplifying LSTM.

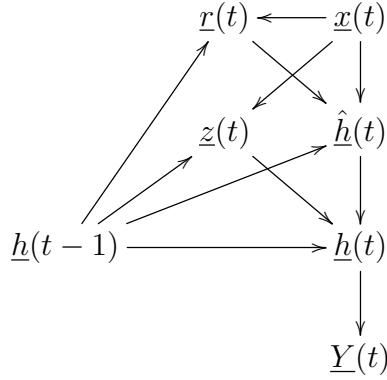


Figure 46.5: bnet for a Gated Recurrent Unit (GRU).

Let

$x(t) \in \mathbb{R}^{nx}$: **input state vector**

$h(t) \in \mathbb{R}^{nh}$: **hidden state vector**

$\hat{h}(t) \in \mathbb{R}^{nh}$: **hidden activation vector**

$z(t) \in \mathbb{R}^{nh}$: **update activation vector**

$r(t) \in \mathbb{R}^{nh}$: **reset activation vector**

$Y(t) \in \mathbb{R}^{ny}$: **classification** of $x(t)$.

$W \in \mathbb{R}^{nh \times nx}$, $U \in \mathbb{R}^{nh \times nh}$ and $b \in \mathbb{R}^{nh}$: weight matrices and bias vectors, parameters learned by training.

$\mathcal{W}^{y|h} \in \mathbb{R}^{ny \times nh}$: weight matrix

Fig.46.5 is a bnet for a GRU. The node TPMs, printed in blue, for this bnet, are as follows.

$$P(z(t)|x(t), h(t-1)) = \mathbb{1}(\quad z(t) = \text{smoid}(W^{z|x}x(t) + U^{z|h}h(t-1) + b^z) \quad), \quad (46.25)$$

where $h(-1) = 0$.

$$P(r(t)|x(t), h(t-1)) = \mathbb{1}(\quad r(t) = \text{smoid}(W^{r|x}x(t) + U^{r|h}h(t-1) + b^r) \quad) \quad (46.26)$$

$$P(\hat{h}(t)|x(t), r(t), h(t-1)) = \mathbb{1}(\quad \hat{h}(t) = \tanh(W^{h|x}x(t) + U^{h|h}(r(t) \odot h(t-1)) + b^h) \quad) \quad (46.27)$$

$$P(h(t)|z(t), h(t-1), \hat{h}(t)) = \mathbb{1}(\quad h(t) = (1 - z(t)) \odot h(t-1) + z(t) \odot \hat{h}(t) \quad) \quad (46.28)$$

$$P(Y(t)|h(t)) = \mathbb{1}(\quad Y(t) = \mathcal{A}(\mathcal{W}^{y|h}h(t) + b^y) \quad) \quad (46.29)$$

Chapter 47

Regression Discontinuity Design

This chapter is based on Ref.[3].

This chapter assumes that the reader has read Chapter 43 on Potential Outcomes (PO).

In Regression Discontinuity Design (RDD), one switches the treatment dose \underline{d} from 0 when $\underline{x} < \xi$ to 1 where $\underline{x} > \xi$, where \underline{x} is an observed confounder (call it the **switch confounder**) and ξ is a threshold value for \underline{x} . One measures the jump δ in the treatment outcome \underline{y} as \underline{x} passes through $\underline{x} = \xi$. Then one makes the very reasonable assumption that δ equals¹ $\mathcal{Y}_{1|x=\xi} - \mathcal{Y}_{0|x=\xi} = ATE|_{x=\xi}$ for an imaginary experiment in which the confounder \underline{x} acts as a normal confounder that doesn't switch the treatment dose \underline{d} .

For example, d^σ might be whether an individual is admitted to Harvard Univ., y^σ might be how much money the individual earns for the first 20 years after graduating from Harvard, and x^σ might be his SAT scores. We assume Harvard only admits students with an SAT score higher than ξ .

47.1 PO analysis

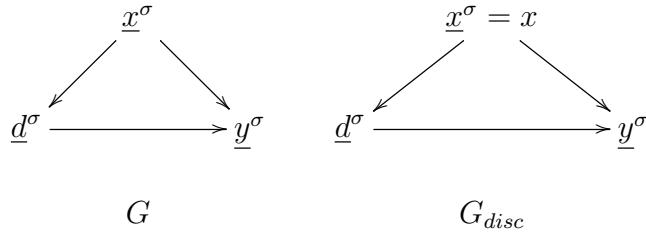


Figure 47.1: 2 bnets used in the PO analysis of RDD. The TPMs for G_{disc} are defined in terms of the TPMs for G . The TPM $P(d^\sigma|x^\sigma)$ for G_{disc} is discontinuous in x^σ .

¹ ATE, which stands for “average treatment effect”, is defined in Chapter 43.

The TPMs, printed in blue, for the bnet G_{disc} shown in Fig.47.1, are as follows. Note that the TPMs for the bnet G_{disc} are defined in terms of the TPMs for the bnet G .

$$P(x^\sigma) = \delta(x^\sigma, x) \quad (47.1)$$

$$P(y^\sigma | d^\sigma, x^\sigma = x) = P_{\underline{y}|\underline{d}, \underline{x}}(y^\sigma | d^\sigma, x) \quad (47.2)$$

$$P(d^\sigma | x^\sigma = x) = \begin{cases} P_{\underline{d}|\underline{x}}(d^\sigma | x^\sigma = x) & \text{for } x > \xi \\ \delta(d^\sigma, 0) & \text{for } x < \xi \end{cases} \quad (47.3)$$

Define

$$E_{\sigma|x}[y^\sigma(d)] = E_{y|x}[y(d)] = \mathcal{Y}_{d|x} \quad (47.4)$$

and

$$\xi \pm = \xi \pm \epsilon \quad (47.5)$$

for some infinitesimal $\epsilon > 0$.

See Fig.47.2. In RDD, we assume that if we define the following 2 δ 's, one for bnet G and the other for bnet G_{disc} , then the two δ 's are equal, and they equal a conditional ATE.

$$\delta_{G_{disc}} = \mathcal{Y}_{1|x=\xi+} - \mathcal{Y}_{0|x=\xi-} \quad (47.6)$$

$$\delta_G = \mathcal{Y}_{1|x=\xi} - \mathcal{Y}_{0|x=\xi} \quad (47.7)$$

$$\delta_G = \delta_{G_{disc}} = \delta \quad (47.8)$$

$$\delta = ATE|_{x=\xi} \quad (47.9)$$

47.2 Linear Regression

In this section, we show how to apply linear regression (LR) to the PO analysis of RDD.

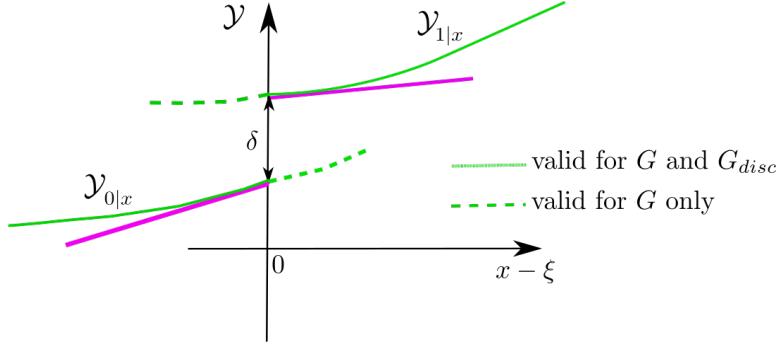


Figure 47.2: The jump δ between $\mathcal{Y}_{1|x}$ and $\mathcal{Y}_{0|x}$ is the same for G and G_{disc} .

y^σ can be fitted as a function of $x \in \mathbb{R}$, for $d^\sigma \in \{0, 1\}$, as follows. Here ϵ^σ is the residual for individual σ and $b_0, m_0, b_1, m_1 \in \mathbb{R}$ are the fit parameters.

$$y^\sigma = [b_0 + m_0(x - \xi)](1 - d^\sigma) + [b_1 + m_1(x - \xi)]d^\sigma + \epsilon^\sigma . \quad (47.10)$$

Note that Eq.(47.10) yields a straight line in the $y^\sigma - x$ plane for $d^\sigma = 0$, and another straight line for $d^\sigma = 1$. These 2 lines are colored magenta in Fig.47.2. We are using the standard symbols b to denote the y-intercept, and m to denote the slope of a straight line.

Taking the expected value of Eq.(47.10), we get

$$\mathcal{Y}_{d|x} = [b_0 + m_0(x - \xi)](1 - d) + [b_1 + m_1(x - \xi)]d . \quad (47.11)$$

Hence,

$$\mathcal{Y}_{1|x=\xi+} = b_1 , \quad \mathcal{Y}_{0|x=\xi-} = b_0 \quad (47.12)$$

$$\delta = b_1 - b_0 \quad (47.13)$$

Chapter 48

Reinforcement Learning (RL)

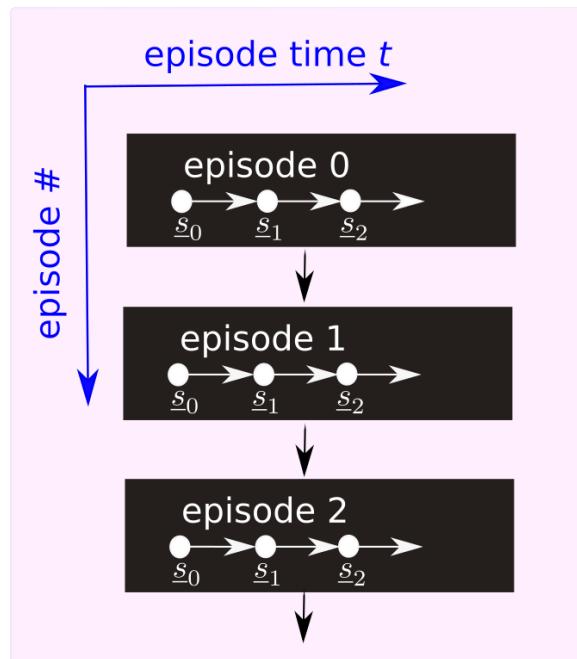


Figure 48.1: Axes for episode time and episode number.

I based this chapter on the following references. Refs.[7][15]

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 $t = 0, 1, \dots, T - 1$. We will consider multiple learning episodes. The episode number will evolve from top to bottom. This is illustrated in Fig.48.1.

Let $\underline{s}_t \in S_s$ for $t \in [0, T - 1]_{\mathbb{Z}}$ be random variables that record the **state** of the agent at various times t .

Let $\underline{a}_t \in S_a$ for $t \in [0, T - 1]_{\mathbb{Z}}$ be random variables that record the **action** of the agent at various times t .

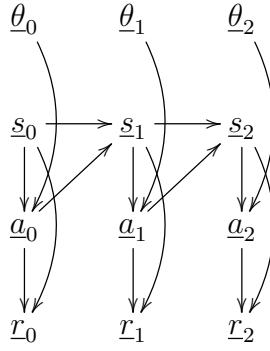


Figure 48.2: State-Action-Reward dynamical bnet

Let $\underline{\theta}_t \in S_\theta$ 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

$$\underline{X}_{\cdot} = [\underline{X}_0, \underline{X}_1, \dots, \underline{X}_{T-1}] . \quad (48.1)$$

Also let

$$\underline{X}_{\geq t} = [\underline{X}_t, \underline{X}_{t+1}, \dots, \underline{X}_{T-1}] . \quad (48.2)$$

Fig.48.2 shows the basic State-Action-Reward bnet for an agent that is learning. The TPMs for the nodes of Fig.48.2 are given in blue below:

$$P(a_t|s_t, \theta_t) = \text{given}. \quad (48.3)$$

$P(a_t|s_t, \theta_t)$ is called a **policy with parameter** θ_t .

$$P(s_t|s_{t-1}, a_{t-1}) = \text{given}. \quad (48.4)$$

$P(s_t|s_{t-1}, a_{t-1})$ is called the **TPM of the model**. $P(s_t|s_{t-1}, a_{t-1})$ reduces to $P(s_0)$ when $t = 0$.

$$P(r_t|s_t, a_t) = \delta(r_t, r(s_t, a_t)) . \quad (48.5)$$

$r : S_s \times S_a \rightarrow \mathbb{R}$ is a given **one-time reward function**.

Note that

$$P(s_{\cdot}, a_{\cdot} | \theta_{\cdot}) = \prod_{t=0}^{T-1} \{P(s_t|s_{t-1}, a_{t-1})P(a_t|s_t, \theta_t)\} . \quad (48.6)$$

Define the **all times reward** Σ by

$$\Sigma(s_{\cdot}, a_{\cdot}) = \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) . \quad (48.7)$$

Here $0 < \gamma < 1$. γ , called the **discount rate**, is included to assure convergence of Σ when $T \rightarrow \infty$. If $r(s_t, a_t) < K$ for all t , then $\Sigma < K \frac{1}{1-\gamma}$.

Define the **objective (i.e. goal) function** $E\Sigma(\theta.)$ by

$$E\Sigma(\theta.) = E_{\underline{s}, \underline{a} | \theta.} \Sigma(\underline{s}, \underline{a}) = \sum_{s., a.} P(s., a. | \theta.) \Sigma(s., a.) \quad (48.8)$$

The goal of RL is to maximize the objective function over its parameters $\theta..$. The parameters θ^* . that maximize the objective function are the optimum strategy:

$$\theta^* = \operatorname{argmax}_{\theta.} E\Sigma(\theta.) \quad (48.9)$$

Define a **future reward** for times $\geq t$ as:

$$\Sigma_{\geq t}((s_{t'}, a_{t'})_{t' \geq t}) = \sum_{t'=t}^{T-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) \quad (48.10)$$

Define the following **expected conditional future rewards** (rewards for times $\geq t$, conditioned on certain quantities having given values):

$$v_t = v(s_t, a_t; \theta.) = E_{\underline{s}, \underline{a} | s_t, a_t, \theta.} [\Sigma_{\geq t}] \quad (48.11)$$

$$V_t = V(s_t; \theta.) = E_{\underline{s}, \underline{a} | s_t, \theta.} [\Sigma_{\geq t}] = E_{\underline{a}_t | s_t, \theta.} [v(s_t, \underline{a}_t; \theta.)] \quad (48.12)$$

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 $v(s_t, a_t; \theta.)$.

Since $E\Sigma_{\geq t}$ only depends on $\theta_{\geq t}$, $v(s_t, a_t; \theta.) = v(s_t, a_t; \theta_{\geq t})$, and $V(s_t; \theta.) = V(s_t; \theta_{\geq t})$.

Note that the objective function $E\Sigma$ can be expressed in terms of v_0 by averaging over its unaveraged parameters:

$$E\Sigma(\theta.) = E_{\underline{s}_0, \underline{a}_0 | \theta_0} v(\underline{s}_0, \underline{a}_0; \theta.) \quad (48.13)$$

Define a **one-time reward** and an **expected conditional one-time reward** as:

$$r_t = r(s_t, a_t) \quad (48.14)$$

$$R_t = R(s_t; \theta_t) = E_{\underline{a}_t | s_t, \theta_t} [r(s_t, \underline{a}_t)] . \quad (48.15)$$

Note that

$$\Sigma_{\geq t} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^{T-1-t} r_{t+(T-1-t)} \quad (48.16)$$

$$= r_t + \gamma \Sigma_{\geq t+1} ; . \quad (48.17)$$

If we take $E_{\underline{s}, \underline{a} | s_t, a_t, \theta}[\cdot]$ of both sides of Eq.(48.17), we get

$$v_t = r_t + \gamma E_{\underline{s}_{t+1}, \underline{a}_{t+1} | \theta} [v_{t+1}] . \quad (48.18)$$

If we take $E_{\underline{s}, \underline{a} | s_t, \theta}[\cdot]$ of both sides of Eq.(48.17), we get

$$V_t = R_t + \gamma E_{\underline{s}_{t+1} | \theta} [V_{t+1}] . \quad (48.19)$$

Note that

$$\Delta r_t = r_t - R_t \quad (48.20)$$

$$= r_t - (V_t - \gamma E_{\underline{s}_{t+1} | \theta} [V_{t+1}]) \quad (48.21)$$

$$= r_t + \gamma E_{\underline{s}_{t+1} | \theta} [V_{t+1}] - V_t . \quad (48.22)$$

Define

$$\Delta v_t = v_t - V_t . \quad (48.23)$$

Note that

$$\Delta v_t = \Delta r_t . \quad (48.24)$$

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)

48.1 Exact RL bnet

An exact RL bnet is given by Fig.48.3.

Fig.48.3 is the same as Fig.48.2 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.48.2.

$$P(\theta_t | \theta) = \delta(\theta_t, (\theta)_t) \quad (48.25)$$

$$\forall(s_t, a_t) : P(v_t(s_t, a_t) | r_t, v_{t+1}(\cdot), \theta) = \delta(v_t(s_t, a_t), r_t + \gamma E_{\underline{s}_{t+1}, \underline{a}_{t+1} | \theta} [v_{t+1}]) \quad (48.26)$$

$$P(\theta' | \theta, v_0(\cdot)) = \delta(\theta', \theta + \alpha \partial_\theta \underbrace{E_{\underline{s}_0, \underline{a}_0 | \theta_0} v(\underline{s}_0, \underline{a}_0; \theta)}_{E\Sigma(\theta)}) \quad (48.27)$$

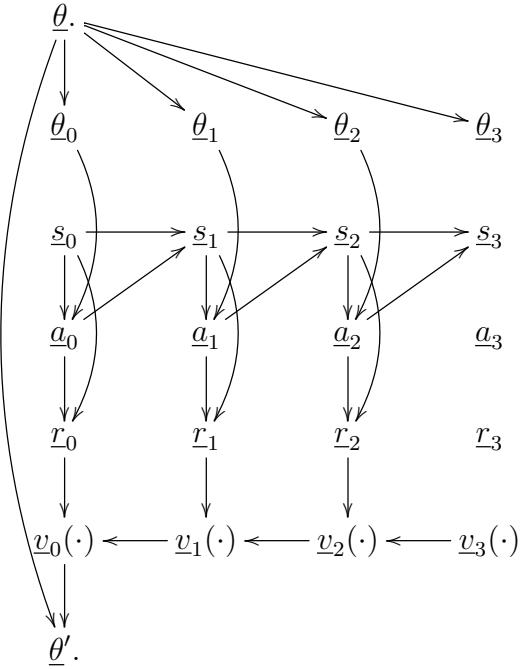


Figure 48.3: Exact RL bnet. $v_t(\cdot)$ means the array $[v_t(s_t, a_t)]_{\forall s_t, a_t}$ The following arrows are implicit: for all t , arrow from $\underline{\theta}.$ $\rightarrow \underline{v}_t(\cdot).$ We did not draw those arrows so as not to clutter the diagram.

$\alpha > 0$ is called the **learning rate**. This method of improving θ . is called gradient ascent.

Concerning the gradient of the objective function, note that

$$\partial_{\theta_t} E\Sigma(\theta.) = \sum_{s., a.} \partial_{\theta_t} P(s., a. | \theta.) \Sigma(s., a.) \quad (48.28)$$

$$= \sum_{s., a.} P(s., a. | \theta.) \partial_{\theta_t} \ln P(s., a. | \theta.) \Sigma(s., a.) \quad (48.29)$$

$$= E_{s., a. | \theta.} \{ \partial_{\theta_t} \ln P(a_t | s_t, \theta_t) \Sigma(s., a.) \} . \quad (48.30)$$

If we run the agent $nsam(\vec{s}_t)$ times and obtain samples $s_t[i], a_t[i]$ for all t and for $i = 0, 1, \dots, nsam(\vec{s}_t) - 1$, we can express this gradient as follows:

$$\partial_{\theta_t} E\Sigma(\theta.) \approx \frac{1}{nsam(\vec{s}_t)} \sum_i \sum_{t=0}^{T-1} \partial_{\theta_t} \ln P(a_t[i] | s_t[i], \theta_t) r(s_t[i], a_t[i]) . \quad (48.31)$$

The exact RL bnet Fig.48.3 is difficult to use to calculate the optimum parameters θ^* . The problem is that s_t propagates towards the future and the $v_t(\cdot)$ 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.

48.2 Actor-Critic RL bnet

For the actor-critic RL bnet, we approximate Eq.(48.31) by

$$\partial_{\theta_t} E\Sigma(\theta.) \approx \frac{1}{nsam(\vec{s})} \sum_i \sum_{t=0}^{T-1} \underbrace{\partial_{\theta_t} \ln P(a_t[i] | s_t[i], \theta_t)}_{Actor} \underbrace{\Delta r_t(s_t[i], a_t[i])}_{Critic} \quad (48.32)$$

The actor-critic RL bnet is given by Fig.48.4. This bnet is approximate and assumes that the policy is known. The TPMs for its nodes are given in blue below.

$$P(\theta_t) = \text{given} \quad (48.33)$$

$$P(s_t[i] | s_{t-1}[i], a_{t-1}[i]) = \text{given} \quad (48.34)$$

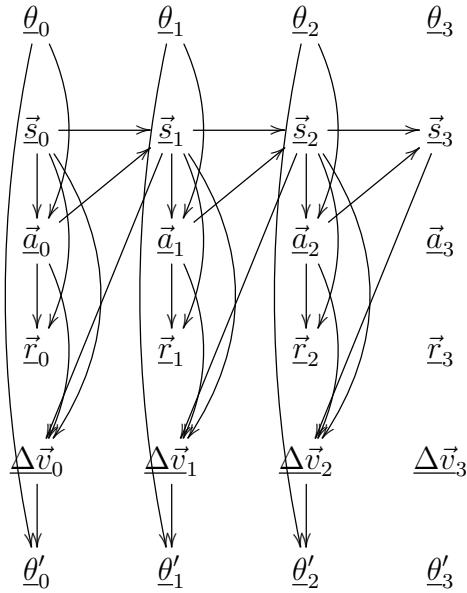


Figure 48.4: Actor-Critic RL bnet.

$$P(a_t[i] | s_t[i], \theta_t) = \text{ given} \quad (48.35)$$

$$P(r_t[i] | s_t[i], a_t[i]) = \delta(r_t[i], r(s_t[i], a_t[i])) \quad (48.36)$$

$r : S_{\underline{s}} \times S_{\underline{a}} \rightarrow \mathbb{R}$ is given.

$$P(\Delta v_t[i] | s_t[i], a_t[i], s_{t+1}[i]) = \delta(\Delta v_t[i], r(s_t[i], a_t[i]) + \gamma \hat{V}(s_{t+1}[i]; \phi') - \hat{V}(s_t[i]); \phi) . \quad (48.37)$$

$$P(\theta'.) = \delta(\theta'., \theta_t + \alpha \partial_{\theta_t} \sum_i \ln P(a_t[i] | s_t[i], \theta_t) \Delta v_t[i]) \quad (48.38)$$

$\hat{V}(s_t[i]; \phi)$ is obtained by curve fitting (see Chapter 5) using samples $(s_t[i], a_t[i])$
 $\forall t, i$ with

$$y[i] = \sum_{t'=t}^T r(s_{t'}[i], a_{t'}[i]) \quad (48.39)$$

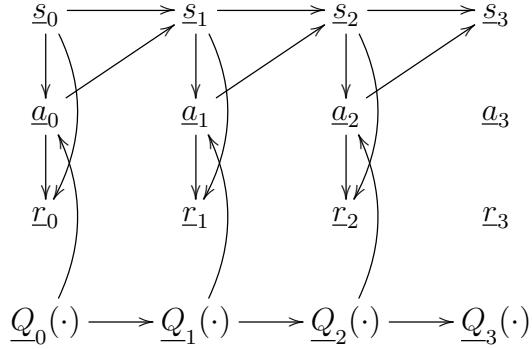


Figure 48.5: Q function learning RL bnet.

and

$$\hat{y}[i] = \hat{V}(s_t[i]; \phi) . \quad (48.40)$$

Eq.(48.39) 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' .

48.3 Q function learning RL bnet

The Q-function learning RL bnet is given by Fig.48.5. This bnet is approximate and assumes that the policy is NOT known. The TPMs for its nodes are given in blue below. (Remember that $Q = v$).

$$P(s_t | s_{t-1}, a_{t-1}) = \text{given} \quad (48.41)$$

$$P(a_t | s_t, v_t(\cdot)) = \delta(a_t, \underset{a}{\operatorname{argmax}} v_t(s_t, a)) \quad (48.42)$$

$$P(r_t | s_t, a_t) = \delta(r_t, r(s_t, a_t)) \quad (48.43)$$

$r : S_s \times S_a \rightarrow \mathbb{R}$ is given.

$$\begin{aligned} \forall(s_t, a_t) : P(v_t(s_t, a_t) | v_{t-1}(\cdot)) &= \\ &= \delta(v_t(s_t, a_t), r(s_t, a_t) + \gamma \max_a E_{s_{t+1}|s_t, a_t} v_{t-1}(s_{t+1}, a)) \end{aligned} \quad (48.44)$$

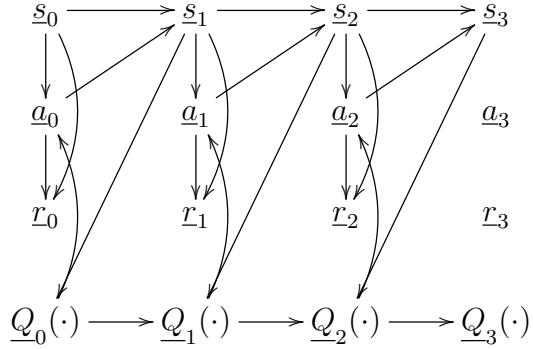


Figure 48.6: Q function learning RL bnet. Same as Fig.48.5 but with new arrow passing s_t to \underline{Q}_{t-1} .

This value for $v_t(s_t, a_t)$ approximates $v_t = r_t + \gamma E_{s_{t+1}, a_{t+1}} v_{t+1}$.

Some people use the bnet of Fig.48.6) instead of Fig.48.5 and replace Eq.(48.44) by

$$\begin{aligned} \forall(s_t, a_t) : P(v_t(s_t, a_t) | s_{t+1}, v_{t-1}(\cdot)) &= \\ &= \delta(v_t(s_t, a_t), r(s_t, a_t) + \gamma \max_a v_{t-1}(s_{t+1}, a)) . \end{aligned} \quad (48.45)$$

Chapter 49

Reliability Box Diagrams and Fault Tree Diagrams

This chapter is based on Refs.[32] and [43].

In this chapter, we assume that reader is familiar with Boolean Algebra. See Chapter Notational Conventions and Preliminaries for a quick review of what we recommend that you know about Boolean Algebra to fully appreciate this chapter.

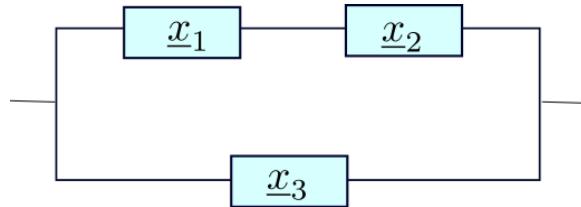


Figure 49.1: Example of rbox diagram.

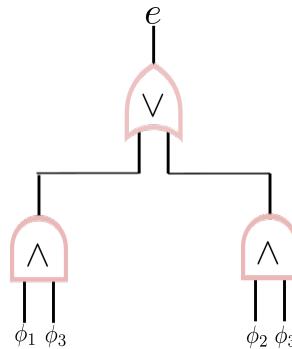


Figure 49.2: An ftree diagram equivalent to Fig.49.1. It represents $e = (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3)$.

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

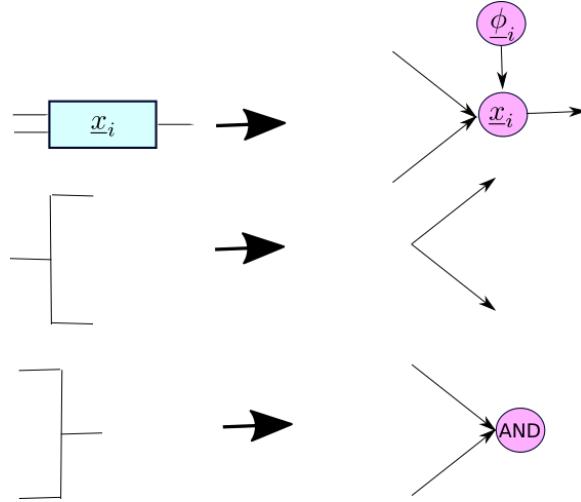


Figure 49.3: How to map an rbox diagram to a bnet.

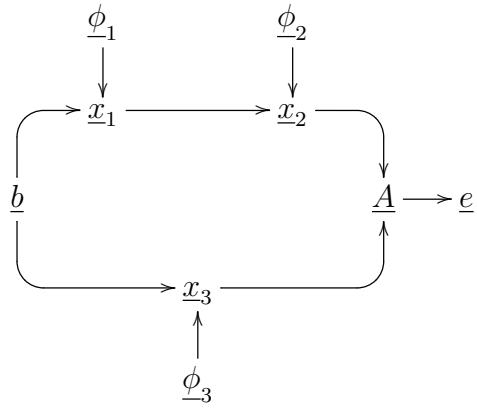


Figure 49.4: bnet corresponding the rbox diagram Fig.49.1.

working in series and one of the 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.49.1 for an example.
- ftree diagrams = Fault Tree Diagrams. See Fig.49.2 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. $\underline{x}^{nx} = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{nx-1})$. $\underline{x}_i \in \{S, F\}$ for all i .

Suppose \underline{x}_i has parents $\underline{\phi}_i$ and $\underline{a}^{na} = (\underline{a}_0, \underline{a}_1, \dots, \underline{a}_{na-1})$. Then the TPM of node \underline{x}_i is defined to be

$$P(\underline{x}_i | \underline{\phi}_i, \underline{a}^{na}) = \delta(\underline{x}_i, \underline{\phi}_i \vee \vee_{i=0}^{na-1} \underline{a}_i) \quad (49.1)$$

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 ,

$$P(\underline{\phi}_i) = \epsilon_i \delta(\underline{\phi}_i, F) + \bar{\epsilon}_i \delta(\underline{\phi}_i, S). \quad (49.2)$$

ϵ_i is the failure probability and $\bar{\epsilon}_i = 1 - \epsilon_i$ 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 $\lambda_i t \ll 1$, where λ_i is the failure rate for node \underline{x}_i and t stands for time. The rblock literature usually calls $\bar{\epsilon}_i = R_i$ 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 $\underline{y}^{ny} = (\underline{y}_0, \underline{y}_1, \dots, \underline{y}_{ny-1})$, then the TPM of \underline{A}_i is

$$P(\underline{A}_i | \underline{y}^{ny}) = \delta(\underline{A}_i, \wedge_{i=0}^{ny-1} \underline{y}_i). \quad (49.3)$$

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.49.3 shows how to translate any rbox diagram to a failure bnet. To illustrate this procedure, we translated the rbox diagram Fig.49.1 into the failure bnet Fig.49.4.

For the failure bnet Fig.49.4, one has:

$$\begin{aligned} P(b) &= \mathbb{1}(b = 0) \\ P(x_1|\phi_1, b) &= \mathbb{1}(x_1 = \phi_1 \vee b) \\ P(x_2|\phi_2, x_1) &= \mathbb{1}(x_2 = \phi_2 \vee x_1) \\ P(x_3|\phi_3, b) &= \mathbb{1}(x_3 = \phi_3 \vee b) \\ P(A|x_2, x_3)e &= \mathbb{1}(x_2 \wedge x_3) \\ P(e|A) &= \mathbb{1}(e = A) \end{aligned} . \quad (49.4)$$

Therefore, all r-instances of this bnet must satisfy

$$e = (\phi_1 \vee \phi_2) \wedge \phi_3 \quad (49.5)$$

$$= (\phi_1 \wedge \phi_3) \vee (\phi_2 \wedge \phi_3) . \quad (49.6)$$

Eq.(49.6) proves that Fig.49.2 is indeed a representation of Fig.49.1.

Next, we consider r-instances of this bnet for two cases: $e = S$ and $e = F$.

- **rblock analysis:** $e = S = 0$.

Table 49.1 shows the probability of all possible r-instances that end in success for the failure bnet Fig.49.4. (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

$$P(e = S) = \bar{\epsilon}_3 + \bar{\epsilon}_1 \bar{\epsilon}_2 \epsilon_3 , \quad (49.7)$$

or, expressing it in reliability language in which $\bar{\epsilon} = R$,

$$P(e = S) = R_3 + R_1 R_2 \bar{R}_3 . \quad (49.8)$$

- **ftree analysis:** $e = F = 1$.

Table 49.2 shows the probability of all possible r-instances that end in failure for the failure bnet Fig.49.4. (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

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 49.1: Probabilities of all possible r-instances with $e = S = 0$ for failure bnet Fig.49.4.

instance	probability
	$\bar{\epsilon}_1 \epsilon_2 \epsilon_3$
	$\epsilon_1 \bar{\epsilon}_2 \epsilon_3$
	$\epsilon_1 \epsilon_2 \epsilon_3$

Table 49.2: Probabilities of all possible r-instances with $e = F = 1$ for the failure bnet Fig.49.4.

cut sets” of the ftree. We will have more to say about minimal cut sets later on. For now, just note from Eq.(49.6) that the ftree Fig.49.2 is just the result of joining together with ORs two expressions, one for each of the two minimal cut sets.

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

$$\text{len}(a^n) = \sum_i a_i \quad (49.9)$$

- **OR gate**

$$P(x_i | \phi_i, a^{na}) = \delta(x_i, \phi_i \vee \vee_j a_j) \quad (49.10)$$

$$= \delta(x_i, \phi_i \vee \mathbb{1}(\text{len}(a^{na}) > 0)) \quad (49.11)$$

- AND gate

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \wedge_j a_j) \quad (49.12)$$

$$= \delta(x_i, \phi_i \vee \mathbb{1}(\text{len}(a^{na}) = na)) \quad (49.13)$$

- Fail if least K failures (less than K successes)

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(\text{len}(a^{na}) \geq K)) \quad (49.14)$$

- Fail if less than K failures (at least K successes)

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(\text{len}(a^{na}) < K)) \quad (49.15)$$

- Fail if exactly one failure

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee \mathbb{1}(\text{len}(a^{na}) = 1)) \quad (49.16)$$

This equals an XOR (exclusive OR) gate when $na = 2$.

- General gate

$$f : \{0, 1\}^{na} \rightarrow \{0, 1\}$$

$$P(x_i|\phi_i, a^{na}) = \delta(x_i, \phi_i \vee f(a^{na})) \quad (49.17)$$

49.1 Minimal Cut Sets

Suppose $x \in \{0, 1\}$ and $f : \{0, 1\} \rightarrow \{0, 1\}$. Then by direct evaluation, we see that

$$f(x) = [\bar{x}f(0)] \vee [xf(1)]. \quad (49.18)$$

Let

$$\begin{aligned} !x &= 1 - x, \\ !^0 x &= x, \\ !^1 x &= !x \end{aligned} \quad (49.19)$$

Then Eq.49.18 can be rewritten as

$$f(x) = \vee_{a \in \{0,1\}} [(\bar{a}x)f(a)] . \quad (49.20)$$

Now suppose $x^n \in \{0, 1\}^n$ and $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Eq.(49.20) generalizes to

$$f(x^n) = \vee_{a^n \in \{0,1\}^n} [\prod_i (\bar{a}_i x_i) f(a^n)] . \quad (49.21)$$

Eq.(49.21) 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.(49.21), 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 $f : \{0, 1\}^{nx} \rightarrow \{0, 1\}$ such that $e = f(\phi^{nx})$ for all the r-instances. We did that for our example failure bnet and obtained Eq.(49.6). 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 Notational Conventions and Preliminaries) to remove those redundancies.

Chapter 50

Restricted Boltzmann Machines

In what follows, we will abbreviate "restricted Boltzmann machine" by rebo.

Let

$$v \in \{0, 1\}^{nv}$$

$$h \in \{0, 1\}^{nh}$$

$b \in \mathbb{R}^{nv}$ (mnemonic, v and b sound the same)

$$a \in \mathbb{R}^{nh}$$

$$W^{v|h} \in \mathbb{R}^{nv \times nh}$$

Energy:

$$E(v, h) = -(b^T v + a^T h + v^T W^{v|h} h) \quad (50.1)$$

Boltzmann distribution:

$$P(v, h) = \frac{e^{-E(v, h)}}{Z} \quad (50.2)$$

Partition function:

$$Z = \sum_{v, h} e^{-E(v, h)} = Z(a, b, W^{v|h}) \quad (50.3)$$

$$P(v|h) = \frac{e^{b^T v + a^T h + v^T W^{v|h} h}}{\sum_v e^{b^T v + a^T h + v^T W^{v|h} h}} \quad (50.4)$$

$$= \frac{e^{b^T v + v^T W^{v|h} h}}{\sum_v e^{b^T v + v^T W^{v|h} h}} \quad (50.5)$$

$$= \prod_i \frac{e^{v_i(b_i + \sum_j W_{i,j}^{v|h} h_j)}}{\sum_{v_i=0,1} e^{v_i(b_i + \sum_j W_{i,j}^{v|h} h_j)}} \quad (50.6)$$

$$= \prod_i P(v_i|h) \quad (50.7)$$

$$P(v_i|h) = \frac{e^{v_i(b_i + \sum_j W_{i,j}^{v|h} h_j)}}{Z_i(h)} \quad (50.8)$$

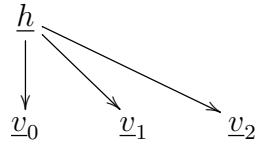


Figure 50.1: bnet for a Restricted Boltzmann Machine (rebo) with $nv = 3$

Eq.50.8 implies that a rebo can be represented by the bnet Fig.50.1.

Let

$$x_i = b_i + \sum_j W_{ij}^{vh} h_j . \quad (50.9)$$

Then

$$P(v_i = 1|h) = \frac{e^{x_i}}{1 + e^{x_i}} \quad (50.10)$$

$$= \frac{1}{1 + e^{-x_i}} \quad (50.11)$$

$$= \text{smoid}(x_i) . \quad (50.12)$$

One could also expand the node h in Fig.50.1 into nh 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 38) with $v_i, h_i \in \{0, 1\}$ and specific $P(h)$ and $P(v_i|h)$ node matrices.

Chapter 51

ROC curves

This chapter is based on Ref.[86].

ROC stands for **Receiver Operating Characteristic**. ROC curves are used in binary classification (BC).

To do BC, we are given the value $x \in \mathbb{R}$ for an individual. From this, we want to decide whether that individual has $a = 0$ or $a = 1$. The decision will depend on the value of a **threshold parameter** $\tau \in \mathbb{R}$.

$$\underline{x} \longleftarrow \underline{a}$$

Figure 51.1: bnet for BC.

Fig.51.1 shows the bnet used for BC.

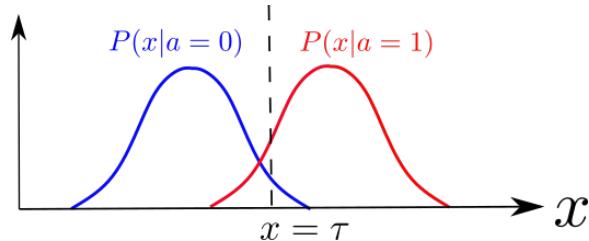


Figure 51.2: x -distribution for two hypotheses $a = 0, 1$.

Fig.51.2 is a plot of $P(x|a)$, i.e., the TPM for node \underline{x} of the bnet in Fig.51.1. Whereas a is binary, x is continuous. But we can replace x by a binary variable

$$b = \mathbb{1}(x > \tau). \quad (51.1)$$

$P(b|a)$ for $b, a \in \{0, 1\}$ is called the **confusion matrix** or **contingency table** for BC. The confusion matrix can be calculated from the TPM $P(x|a)$. Fig.51.3 illustrates

		Actual Value (a)	
		0	1
Predicted Value (x)	0	$TNR \triangleleft$	$FNR \wedge$
	1	$FPR \wedge$	$TPR \triangleleft$

Figure 51.3: The confusion matrix $P(b|a)$ for BC.

the confusion matrix $P(b|a)$ for BC. In that figure, the rates R are defined as follows.¹

- **True Negative Rate (TNR)**

$$R_{0|0}(\tau) = P(x < \tau | a = 0) = \int_{x < \tau} dx P(x | a = 0) \quad (51.2)$$

- **False Positive Rate (FPR)**

$$R_{1|0}(\tau) = 1 - R_{0|0}(\tau) \quad (51.3)$$

$$= P(x > \tau | a = 0) = \int_{x > \tau} dx P(x | a = 0) \quad (51.4)$$

In Hypothesis Testing, $R_{1|0}$ is called the **p-value that $x > \tau$ assuming curve 0 is the null hypothesis**.

- **False Negative Rate (FNR)**

$$R_{0|1}(\tau) = P(x < \tau | a = 1) = \int_{x < \tau} dx P(x | a = 1) \quad (51.5)$$

In Hypothesis Testing, $R_{0|1}$ is called the **p-value that $x < \tau$ assuming curve 1 is the null hypothesis**.

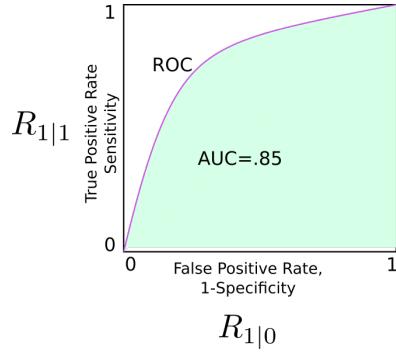
- **True Positive Rate (TPR)**

$$R_{1|1}(\tau) = 1 - R_{0|1}(\tau) \quad (51.6)$$

$$= P(x > \tau | a = 1) = \int_{x > \tau} dx P(x | a = 1) \quad (51.7)$$

The **Receiver Operating Characteristic (ROC)** is a parametric plot with $X = R_{1|0}(\tau)$ and $Y = R_{1|1}(\tau)$, where $\tau \in \mathbb{R}$. The **Area Under the Curve (AUC)** is the area under the ROC. Fig.51.4 shows an example of a ROC and its AUC.

¹ I find the notation $x|a$ where $x, a \in \{0, 1\}$ much clearer than $\alpha\beta$ where $\alpha = T, F$ and $\beta = N, P$. Note that $\alpha = \mathbb{1}(x = a)$ and $\beta = x$, if we identify $0 = F = N$ and $1 = T = P$.



$$R_{1|0}$$

Figure 51.4: Example of ROC. Green shaded area is the AUC of the ROC.

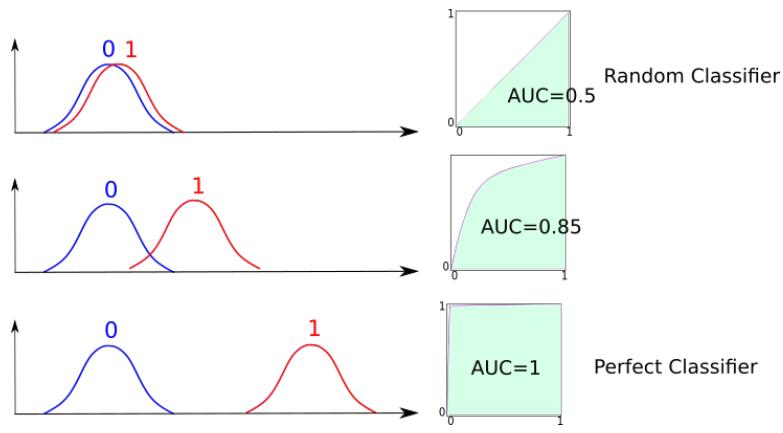


Figure 51.5: ROC curves for 3 different separations between the 0 and 1 x-distributions.

Fig.51.5 shows situations that give $AUC=.5$ (random classifier), $AUC=.85$, and $AUC=1$ (perfect classifier). It's also possible to get an $AUC \in [0, 0.5]$, but we will ignore those models because they are useless for BC.

Note that

$$AUC = \int_{x=0}^1 d\tau R_{1|1}(\tau) \frac{dR_{1|0}(\tau)}{d\tau} \quad (51.8)$$

$$= \int_{-\infty}^{-\infty} d\tau \left\{ \int_{-\infty}^{\infty} dx \mathbb{1}(x > \tau) P(x|a=1) \right\} (-1) P(x = \tau|a=0) \quad (51.9)$$

$$= \int_{-\infty}^{\infty} dx' \int_{-\infty}^{\infty} dx \mathbb{1}(x > x') P(x|a=1) P(x'|a=0) . \quad (51.10)$$

51.1 Terminology Table Adapted from Wikipedia Ref.[86]

Let $N_{x|a}$ be numbers (counts) so that

$$P(x|a) = \frac{N_{x|a}}{\sum_{x'} N_{x'|a}} \quad (51.11)$$

for all $x, a \in \{0, 1\}$.

condition positive (P): $\sum_x N_{x|1}$, the number of real positive cases in the data

condition negative (N): $\sum_x N_{x|0}$, the number of real negative cases in the data

true positive (TP): $N_{1|1}$, hit

true negative (TN): $N_{0|0}$, correct rejection

false positive (FP): $N_{1|0}$, false alarm, type I error or underestimation

false negative (FN): $N_{0|1}$, miss, type II error or overestimation

sensitivity, recall, hit rate, or true positive rate (TPR):

$$TPR = R_{1|1} = \frac{N_{1|1}}{P} = \frac{N_{1|1}}{N_{1|1} + N_{0|1}} = 1 - R_{0|1} \quad (51.12)$$

specificity, selectivity or true negative rate (TNR):

$$TNR = R_{0|0} = \frac{N_{0|0}}{N} = \frac{N_{0|0}}{N_{0|0} + N_{1|0}} = 1 - R_{1|0} \quad (51.13)$$

precision or positive predictive value (PPV):

$$PPV = \frac{N_{1|1}}{N_{1|1} + N_{1|0}} = 1 - FDR \quad (51.14)$$

negative predictive value (NPV):

$$NPV = \frac{N_{0|0}}{N_{0|0} + N_{0|1}} = 1 - FOR \quad (51.15)$$

miss rate or false negative rate (FNR):

$$FNR = R_{0|1} = \frac{N_{0|1}}{P} = \frac{N_{0|1}}{N_{0|1} + N_{1|1}} = 1 - R_{1|1} \quad (51.16)$$

fall-out or false positive rate (FPR):

$$FPR = R_{1|0} = \frac{N_{1|0}}{N} = \frac{N_{1|0}}{N_{1|0} + N_{0|0}} = 1 - R_{0|0} \quad (51.17)$$

false discovery rate (FDR):

$$FDR = \frac{N_{1|0}}{N_{1|0} + N_{1|1}} = 1 - PPV \quad (51.18)$$

false omission rate (FOR):

$$FOR = \frac{N_{0|1}}{N_{0|1} + N_{0|0}} = 1 - NPV \quad (51.19)$$

accuracy (ACC):

$$ACC = \frac{N_{1|1} + N_{0|0}}{P + N} = \frac{N_{1|1} + N_{0|0}}{N_{1|1} + N_{0|0} + N_{1|0} + N_{0|1}} \quad (51.20)$$

balanced accuracy (BA):

$$BA = \frac{R_{1|1} + R_{0|0}}{2} \quad (51.21)$$

F1 score is the harmonic mean of precision and sensitivity:

$$F_1 = 2 \times \frac{PPV \times R_{1|1}}{PPV + R_{1|1}} = \frac{2N_{1|1}}{2N_{1|1} + N_{1|0} + N_{0|1}} \quad (51.22)$$

Chapter 52

Scoring the Nodes of a Learned Bnet

Chapter 54 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 $\vec{x} = (x[\sigma])_{\sigma=0,1,\dots,nsam-1}$, where $x[\sigma] \sim P_{\underline{x}|\theta}(x|\theta)$:

$$\text{ML-score} = \ln(P(\vec{x}|\theta)) \quad (52.1)$$

$$= \ln \prod_{\sigma} P(x[\sigma] | \theta) \quad (52.2)$$

$$= \sum_{\sigma} \ln P(x[\sigma] | \theta) \quad (52.3)$$

$$\approx nsam \sum_x P(x|\theta) \ln P(x|\theta) \quad (52.4)$$

$$= -nsam H(P_{\underline{x}|\theta}) , \quad (52.5)$$

and SIT scores consist of a negative entropy:

$$\text{Info-score} = -H(P_{\underline{x}|\theta}) . \quad (52.6)$$

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 .

52.1 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.[51]
- Multinomial Distribution, Ref.[78]
- Dirichlet Distribution, Ref.[55]
- Multivariate Normal Distribution, Ref.[80]
- Beta function, Ref.[47]
- Multinomial Coefficients, Ref.[79]
- Gamma Function Ref.[58]

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 $q_\cdot = (q_0, q_1, \dots, q_{nq-1})$ for various quantities q

Gamma function. If $n > 0$ is an integer,

$$\Gamma(n+1) = n! \quad (52.7)$$

The **multivariate Beta function** is defined by

$$B(\alpha_\cdot) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\alpha_+)} \quad (52.8)$$

where $\alpha_k > 0$ for all k .

The **multinomial coefficient** is defined by

$$C(N_\cdot) = \frac{N_+!}{\prod_k N_k!} \quad (52.9)$$

where N_k are non-negative integers.

The **inverse of the multinomial coefficient** will be denoted by

$$CI(N_\cdot) = \frac{1}{C(N_\cdot)} = \frac{\prod_k N_k!}{N_+!} \quad (52.10)$$

The **Categorical Distribution** is defined by

$$Cat(x; \pi.) = \pi_x = \prod_k \pi_k^{\mathbb{1}(k=x)} \quad (52.11)$$

for $k, x \in S_x$, where $\pi.$ is a probability dist. (i.e., $\pi_k \geq 0$ for all k , and $\pi_+ = 1$).

The **Multinomial Distribution** is defined by

$$Mul(N.; \pi., N) = C(N.) , \quad (52.12)$$

where N_k is a non-negative integer for all k , $N_+ = N$, and $\pi.$ is a probability dist. $Mul()$ satisfies:

$$E[N_k] = N\pi_k . \quad (52.13)$$

The **Dirichlet Distribution** is defined by

$$Dir(\pi.; \alpha.) = \frac{1}{B(\alpha.)} \prod_k \pi_k^{\alpha_k - 1} \quad (52.14)$$

where $\alpha_k > 0$ for all k , and $\pi.$ is a probability dist. The $\alpha.$ are called **concentration parameters** or **hyperparameters**. $Dir()$ satisfies:

$$E[\pi_k] = \frac{N_k}{N_+} . \quad (52.15)$$

$Dir()$ is **conjugate prior** of $Mul()$

Note that

$$Mul(N.; \pi., N) Dir(\pi.; \alpha.) = \mathcal{K}(N., \alpha.) Dir(\pi.; N. + \alpha.) , \quad (52.16)$$

where

$$\mathcal{K}(N., \alpha.) = \frac{B(N. + \alpha.)}{CI(N.)B(\alpha.)} . \quad (52.17)$$

$Dir()$ 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

$$Dir(\pi.; \alpha_k = N_k + 1) = C(N.) \prod_k \pi_k^{N_k} \quad (52.18)$$

$$= Mul(N.; \pi., N_+) . \quad (52.19)$$

52.2 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 $k \in \{0, 1, \dots, nk - 1\}$ of a single node \underline{x} , let

\underline{N}_k = current count number (an integer, data)

$\underline{\pi}_.$ = a probability dist, the TPM for the node

$\underline{\alpha}_k$ = prior count number

$$\underline{N}_. \longleftarrow \underline{\pi}_. \longleftarrow \underline{\alpha}_.$$

Figure 52.1: 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 ($\underline{\alpha}_.$) .

Consider the Markov chain bnet of Fig.52.1, with the following TPMs, given in blue.

$$P(N.| \underline{\pi}_.) = \text{Mul}(N.; \underline{\pi}_., N_+) \quad (52.20)$$

$$P(\underline{\pi}_.| \underline{\alpha}_.) = \text{Dir}(\underline{\pi}_.; \underline{\alpha}_.) \quad (52.21)$$

It follows that

$$P(N., \underline{\pi}_.| \underline{\alpha}_.) = P(N.| \underline{\pi}_.)P(\underline{\pi}_.| \underline{\alpha}_.) \quad (52.22)$$

$$= \text{Mul}(N.; \underline{\pi}_., N_+) \text{Dir}(\underline{\pi}_.; \underline{\alpha}_.) \quad (52.23)$$

$$= \mathcal{K}(N., \underline{\alpha}_.) \text{Dir}(\underline{\pi}_.; N. + \underline{\alpha}_.). \quad (52.24)$$

From Eq.(52.15) for the expected value of $\text{Dir}()$, we get

$$\hat{\pi}_. = E[\underline{\pi}_.] = \frac{N. + \underline{\alpha}_.}{N_+ + \underline{\alpha}_+}. \quad (52.25)$$

Integrating both sides of Eq.(52.24) over $\underline{\pi}_.$, we find that

$$P(N.| \underline{\alpha}_.) = \mathcal{K}(N., \underline{\alpha}_.). \quad (52.26)$$

If $N_k \gg 1$ for all k , then the $\text{Dir}()$ in Eq.(52.24) can be replaced by a $\text{Mul}()$

$$P(N., \underline{\pi}_.| \underline{\alpha}_.) \approx \mathcal{K}(N., \underline{\alpha}_.) \text{Mul}(N. + \underline{\alpha}_.; \underline{\pi}_., N_+ + \underline{\alpha}_+). \quad (52.27)$$

Therefore,

$$P(N.| \pi., \alpha.) = \frac{P(N., \pi.| \alpha.)}{P(N.| \alpha.)} \quad (52.28)$$

$$= Mul(N. + \alpha.; \pi., N_+ + \alpha_+) . \quad (52.29)$$

Claim 29

$$\ln P(N.| \hat{\pi}., \alpha.) = -(N_+ + \alpha_+)H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) + \ln C(N. + \alpha.) \quad (52.30)$$

$$> -(N_+ + \alpha_+)H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) - \frac{1}{2}(nk - 1)\ln N_+ \quad (52.31)$$

proof:

$$\ln P(N.| \hat{\pi}., \alpha.) = \sum_k (N_k + \alpha_k) \ln \hat{\pi}_k + \ln C(N. + \alpha.) \quad (52.32)$$

$$= \sum_k (N_k + \alpha_k) \ln \frac{N_k + \alpha_k}{N_+ + \alpha_+} + \ln C(N. + \alpha.) \quad (52.33)$$

$$= -(N_+ + \alpha_+)H\left(\frac{N. + \alpha.}{N_+ + \alpha_+}\right) + \ln C(N. + \alpha.) \quad (52.34)$$

Recall Stirling's approximation of a factorial, valid for large integers n :

$$\ln n! \approx (n + \frac{1}{2}) \ln n - n . \quad (52.35)$$

Assume $N_k \gg 1$ for all k . Applying Stirling's approximation to all factorials in $C(N)$, we get

$$\ln C(N.) \approx (N_+ + \frac{1}{2}) \ln N_+ - N_+ - \sum_k \left[(N_k + \frac{1}{2}) \ln N_k - N_k \right] \quad (52.36)$$

$$= (N_+ + \frac{1}{2}) \ln N_+ - \sum_k (N_k + \frac{1}{2}) \ln N_k . \quad (52.37)$$

Next assume that

$$N_k \approx \frac{N_+}{nk} . \quad (52.38)$$

Then

$$\ln C(N.) = (N_+ + \frac{1}{2}) \ln N_+ - nk(\frac{N_+}{nk} + \frac{1}{2})[\ln N_+ - \ln nk] \quad (52.39)$$

$$= -\frac{1}{2}(nk - 1) \ln N_+ + (N_+ + \frac{nk}{2}) \ln nk \quad (52.40)$$

$$> -\frac{1}{2}(nk - 1) \ln N_+ . \quad (52.41)$$

QED

52.3 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 previous section are valid for the general case if we make the following replacements: $\pi.$ $\rightarrow \underline{\pi^i}_{|\mu}$ $N.$ $\rightarrow \underline{N^i}_{,\mu}$ $\alpha.$ $\rightarrow \underline{\alpha^i}_{,\mu}$. Upon this replacement, Fig.52.1 becomes Fig.52.2. The TPMs, printed in blue, of the new Markov chain, are as follows:

$$\underline{N^i}_{,\mu} \longleftarrow \underline{\pi^i}_{|\mu} \longleftarrow \underline{\alpha^i}_{,\mu}$$

Figure 52.2: Generalization of Fig.52.1. For a bnet with multiple nodes each of which may have multiple parents, this is a Markov chain of current counts ($\underline{N^i}_{,\mu}$), TPM ($\underline{\pi^i}_{|\mu}$), and prior counts ($\underline{\alpha^i}_{,\mu}$) .

$$P(\underline{N^i}_{,\mu} | \underline{\pi^i}_{|\mu}) = Mul(\underline{N^i}_{,\mu}; \underline{\pi^i}_{|\mu}, \underline{N^i}_{+, \mu}) \quad (52.42)$$

$$P(\underline{\pi^i}_{|\mu} | \underline{\alpha^i}_{,\mu}) = Dir(\underline{\pi^i}_{|\mu}; \underline{\alpha^i}_{,\mu}) \quad (52.43)$$

In these TPMs,

$i \in S_i = \{0, 1, \dots, ni - 1\}$ = node index

$\underline{x}.$ = $(\underline{x}_i)_{i \in S_i}$ = the nodes of the learned bnet.

$k \in S_k^i = \{0, 1, \dots, nk^i - 1\}$ = states of node x_i

$\mu \in S_\mu^i = \{0, 1, \dots, n\mu^i - 1\}$ = 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:

$$|B| = \sum_i (nk^i - 1)n\mu^i . \quad (52.44)$$

Suppose that we are given $nsam$ samples $\vec{x}_i = (\underline{x}_i[\sigma])_{\sigma=0,1,\dots,nsam-1}$ of our learned bnet. The count numbers $N_{k,\mu}^i$ are defined in terms of those samples as follows:

$$N_{k,\mu}^i = \sum_{\sigma} \mathbb{1}(\underline{x}_i[\sigma] = k, pa(\underline{x}_i[\sigma]) = \mu) . \quad (52.45)$$

It is also convenient to defined count number ratios

$$N_{k|\mu}^i = \frac{N_{k,\mu}^i}{N_{+,\mu}^i} . \quad (52.46)$$

Note that $N_{k,\mu}^i$ is a positive integer whereas $N_{k|\mu}^i \in [0, 1]$.

Let's denote the components of the TPMs by $\pi_{k|\mu}^i$:

$$\pi_{k|\mu}^i = P(\underline{x}_i = k | pa(\underline{x}_i) = \mu) \approx N_{k|\mu}^i . \quad (52.47)$$

The rest of this section lists equations that we obtained from the previous section, by adding the new indices i, μ :

$$\mathcal{K}(N_{\cdot,\mu}^i, \alpha_{\cdot,\mu}^i) = \frac{B(N_{\cdot,\mu}^i + \alpha_{\cdot,\mu}^i)}{CI(N_{\cdot,\mu}^i)B(\alpha_{\cdot,\mu}^i)} \quad (52.48)$$

$$\hat{\pi}_{k|\mu}^i = \frac{N_{k,\mu}^i + \alpha_{k,\mu}^i}{N_{+,\mu}^i + \alpha_{+,\mu}^i} \quad (52.49)$$

$$P(N_{\cdot,\mu}^i | \alpha_{\cdot,\mu}^i) = \mathcal{K}(N_{\cdot,\mu}^i, \alpha_{\cdot,\mu}^i) \quad (52.50)$$

$$P(N_{\cdot,\mu}^i | \pi_{\cdot|\mu}^i, \alpha_{\cdot,\mu}^i) \approx Mul(N_{\cdot,\mu}^i + \alpha_{\cdot,\mu}^i; \pi_{\cdot|\mu}^i, N_{+,\mu}^i + \alpha_{+,\mu}^i) \quad (52.51)$$

Claim 30

$$\ln P(N_{\cdot,\mu}^i | \hat{\pi}_{\cdot|\mu}^i, \alpha_{\cdot,\mu}^i) = \sum_k (N_{k,\mu}^i + \alpha_{k,\mu}^i) \ln \left(\frac{N_{k,\mu}^i + \alpha_{k,\mu}^i}{N_{+,\mu}^i + \alpha_{+,\mu}^i} \right) + \ln C(N_{\cdot,\mu}^i + \alpha_{\cdot,\mu}^i) \quad (52.52)$$

$$> \sum_k (N_{k,\mu}^i + \alpha_{k,\mu}^i) \ln \left(\frac{N_{k,\mu}^i + \alpha_{k,\mu}^i}{N_{+,\mu}^i + \alpha_{+,\mu}^i} \right) - \frac{1}{2}(nk^i - 1) \ln N_{+,\mu}^i \quad (52.53)$$

52.4 Bayesian Scores

- Bayesian Information Criterion (BIC)

$$\text{BIC-score} = - \sum_i \sum_{k,\mu} N_{k,\mu}^i \ln \left(\frac{N_{k,\mu}^i}{N_{+, \mu}^i} \right) + \underbrace{-\frac{|B|}{2} \ln N_{+,+}^+}_{\sum_i \sum_\mu \ln C(N_{\cdot, \mu}^i) \text{ would be more accurate}} \quad (52.54)$$

$$\approx \sum_i \sum_\mu \ln P(N_{\cdot, \mu}^i | \hat{\pi}_{\cdot, \mu}^i, \alpha_{\cdot, \mu}^i = 0) \quad (52.55)$$

- Bayesian Dirichlet (BD)

$$\text{BD-score} = \sum_i \sum_\mu \ln \frac{B(N_{\cdot, \mu}^i + \alpha_{\cdot, \mu}^i)}{B(N_{\cdot, \mu}^i)} \quad (52.56)$$

$$= \sum_i \sum_\mu \ln [CI(N_{\cdot, \mu}^i) P(N_{\cdot, \mu}^i | \alpha_{\cdot, \mu}^i)] \quad (52.57)$$

- BD equivalent (BDe)

$$\text{BDe-score} = \text{BD-score} (\alpha_{k,\mu}^i = \alpha' N_{k,\mu}^i), \quad (52.58)$$

where α' is a free parameter.

- BD equivalent unified (BDeu)

$$\text{BDeu-score} = \text{BD-score} \left(\alpha_{k,\mu}^i = \frac{\alpha'}{n k^i n \mu^i} \right), \quad (52.59)$$

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 42 for more information about observational equivalence.

52.5 Information Theoretic scores

- Maximum likelihood

$$\text{ML-score} = \sum_i \sum_{k,\mu} N_{k,\mu}^i \ln N_{k|\mu}^i \quad (52.60)$$

$$= - \sum_i H(\underline{k}^i | \underline{\mu}^i), \quad (52.61)$$

where $P_{\underline{k}^i | \underline{\mu}^i}(k | \mu) = N_{k|\mu}^i$ and $P_{\underline{k}^i, \underline{\mu}^i}(k, \mu) = N_{k,\mu}^i$.

- Bayesian Information Criterion (BIC), aka Minimum Description Length (MDL)

$$\text{BIC-score} = \text{ML-score} - \frac{|B|}{2} \ln N_{+,+}^+ \quad (52.62)$$

$$\approx \sum_i \sum_{k,\mu} N_{k,\mu}^i \ln \frac{N_{k|\mu}^i}{\sqrt{N_{+,+}^+}} \quad (52.63)$$

- Akaike Information Criterion (AIC)

$$\text{AIC-score} = \text{ML-score} - |B| \quad (52.64)$$

$$\approx \sum_i \sum_{k,\mu} N_{k,\mu}^i [\ln N_{k|\mu}^i - 1] \quad (52.65)$$

Chapter 53

Simpson's Paradox

This chapter is based on Chapter 6 of “The Book of Why”, Ref.[31]. See also Ref.[89] 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

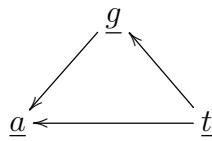


Figure 53.1: bnet for a simple example of Simpson’s paradox. Here node \underline{g} is a chain junction and a mediator.

This situation can be modeled by either bnet Fig.53.1. or bnet Fig.53.2. The two bnets are probabilistically equivalent (i.e., they both represent the same proba-

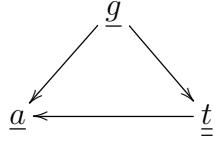


Figure 53.2: bnet that is probabilistically but not physically equivalent to bnet Fig.53.1. Here node \underline{g} is a fork junction and a confounder.

bility distribution $P(a, t, \underline{g})$) because

$$P(g|t)P(t) = P(\underline{g}, t) = P(t|\underline{g})P(\underline{g}) . \quad (53.1)$$

For the bnet Fig.53.1, one has

$$P(a, \underline{g}, t) = P(a|\underline{g}, t)P(\underline{g}|t)P(t) . \quad (53.2)$$

Therefore,

$$P(a = 1|t) = \sum_g P(a = 1|t, \underline{g})P(\underline{g}|t) = E_{\underline{g}|t}P(a = 1|t, \underline{g}) , \quad (53.3)$$

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

$$\vec{q}^g = [P(a = 1|t, \underline{g})]_{t=0,1} \quad (53.4)$$

for $g = 0, 1$, and

$$\vec{q}^* = [P(a = 1|t)]_{t=0,1} . \quad (53.5)$$

It is possible (see Fig.53.3 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.

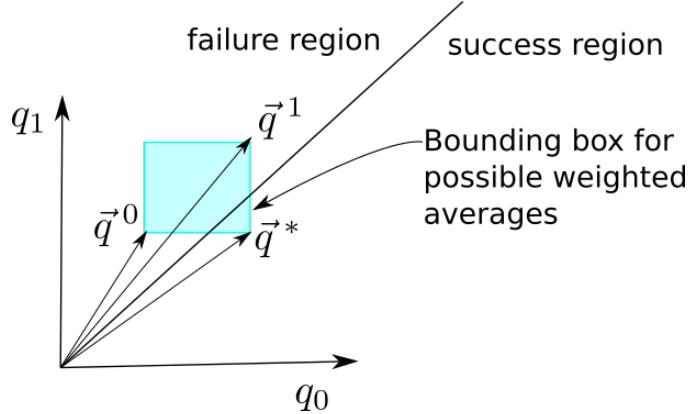


Figure 53.3: \vec{q}^0 , \vec{q}^1 vectors and bounding box for vector \vec{q}^* .

53.1 Pearl Causality

Pearl Causality would add the following two important insights to this problem:

1. bnets Fig.53.1 and Fig.53.2, although they are probabilistically equivalent, do not represent the same physical situation. In fact, only Fig.53.2 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.53.2. 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 $E_{\underline{g}|\underline{t}}$ in our equation for $P(a = 1|\underline{t})$ becomes a simpler average $E_{\underline{g}}$ which is independent of \underline{t} . But for such an average, the bounding box in Fig.53.3 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.

53.2 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 53.1: 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.[31].

$$P(a|t, g) = \begin{array}{c|cccc} & 0,0 & 0,1 & 1,0 & 1,1 \\ \hline 0 & 19/20 & 28/40 & 37/40 & 12/20 \\ 1 & 1/20 & 12/40 & 3/40 & 8/20 \end{array} \quad (53.6)$$

$$P(a|t) = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 47/60 & 49/60 \\ 1 & 13/60 & 11/60 \end{array} \quad (53.7)$$

$$\begin{aligned} \frac{P(a=1,t=1,g=0)}{\sum_a P(a,t=1,g=0)} &= P(a = 1|t = 1, g = 0) &= \frac{3}{40} \\ \frac{P(a=1,t=0,g=0)}{\sum_a P(a,t=0,g=0)} &= P(a = 1|t = 0, g = 0) &= \frac{1}{20} = \frac{2}{40} \end{aligned} \quad (53.8)$$

$$\begin{aligned} \frac{P(a=1,t=1,g=1)}{\sum_a P(a,t=1,g=1)} &= P(a = 1|t = 1, g = 1) &= \frac{8}{20} = \frac{16}{40} \\ \frac{P(a=1,t=0,g=1)}{\sum_a P(a,t=0,g=1)} &= P(a = 1|t = 0, g = 1) &= \frac{12}{40} \end{aligned} \quad (53.9)$$

$$\begin{aligned} \frac{\sum_g P(a=1,t=1,g)}{\sum_g \sum_a P(a,t=1,g)} &= P(a = 1|t = 1) &= \frac{11}{60} \\ \frac{\sum_g P(a=1,t=0,g)}{\sum_g \sum_a P(a,t=0,g)} &= P(a = 1|t = 0) &= \frac{13}{60} \end{aligned} \quad (53.10)$$

Note that the right hand side of Eq.53.8 is higher for $t = 1$ than for $t = 0$. Same trend occurs in Eqs.53.9 but is reversed in Eqs.53.10.

Chapter 54

Structure and Parameter Learning for Bnets

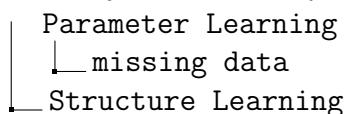
Learning a bnet from data is a computationally intensive NP-complete problem. Therefore, the best one can hope for is for heuristic algorithms that solve this problem approximately. A huge number of such algorithms have been tried and continue to be tried. Luckily, there exists a free open source software 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.[34], and on a 2019 survey paper [35] by Scutari et al. I highly recommend looking at both. Refs. [2] and [16] were also helpful to me in understanding this subject.

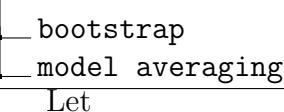
`bnlearn` (Ref.[34]) (free, open source) is very comprehensive and well maintained. It is written mostly in C with an R frontend. It was developed by Marco Scutari and collaborators over a time period of more than 10 years, and is still under active development. How things stand in the field of bnet 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.

54.1 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`.



```
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
```



-
- 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 52, define

$$\pi_{k|\mu}^i = P(\underline{x}_i = k \mid pa(\underline{x}_i) = \mu). \quad (54.1)$$

Then $\pi_{k|\mu}^i$ can be estimated from the data $N_{k,\mu}^i$ using:

$$\pi_{k|\mu}^i \approx N_{k|\mu}^i = \frac{N_{k,\mu}^i}{N_{+,\mu}^i}. \quad (54.2)$$

PL described by Eq.(54.2) 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 36 on missing data and Chapter 20 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.[35] by Scutari et al compares the performance of many different bnet learning algorithms.

54.2 Score based SL algorithms

Score based SL algorithms require scoring bnets (with either all-discrete, all-continuous or mixed nodes). See Chapter 52 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.

54.3 Constraint based SL algorithms

To fully understand constraint based SL algorithms, the reader is advised to read Chapters 16 and 42 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 $\bar{P}(x.|a.)$ 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.

54.4 Pseudo-code for some bnet learning algorithms

Algorithm 2: 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) $T = TPMs(G, D)$.

```

 $E \leftarrow \emptyset$ 
 $T \leftarrow \emptyset$ 
 $B \leftarrow (V, E, T)$ 
 $maxscore \leftarrow -\infty$ 
// 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
                 $T' \leftarrow TPMs(G', D)$ 
                 $B' \leftarrow (G', T')$ 
                 $newscore = \text{BIC-score}(B')$ 
                if  $newscore > maxscore$  then
                     $B \leftarrow B'$ 
                     $maxscore \leftarrow newscore$ 
                else
                     $again \leftarrow False$ 
    return  $B$ 

```

Algorithm 3: 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.

```

 $E \leftarrow \emptyset$ 
// 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.
 $\text{for } \lambda = 0, 1, 2, \dots, |V| - 2 \text{ do}$ 
     $\quad \text{for all } \underline{x} - \underline{y} \in UE \text{ do}$ 
         $\quad \quad \text{for all } S = \{a \in V : \underline{x} - a \in UE, a \neq \underline{x}, \underline{y}\} \ni |S| = \lambda \text{ do}$ 
             $\quad \quad \quad \text{if } H(\underline{x} : \underline{y} | S) < \epsilon \text{ then}$ 
                 $\quad \quad \quad \quad /* \text{ If there were an arrow between } \underline{x} \text{ and } \underline{y}, \text{ then}$ 
                 $\quad \quad \quad \quad \text{conditioning on } S \text{ would not be enough to interrupt}$ 
                 $\quad \quad \quad \quad \text{info transmission } H(\underline{x} : \underline{y} | S) \text{ between } \underline{x} \text{ and } \underline{y} */$ 
                 $\quad \quad \quad \quad UE \leftarrow UE - \{\underline{x} - \underline{y}\}$ 
                 $\quad \quad \quad \quad S(\underline{x} - \underline{y}) \leftarrow S$ 
     $\quad // Growth phase. Adds v structures to E.$ 
     $\quad \text{for all } \underline{x}, \underline{y}, \underline{a} \text{ such that } \underline{x} - \underline{a} \in UE, \underline{a} - \underline{y} \in UE, \underline{x} - \underline{y} \notin UE, \underline{a} \notin S(\underline{x} - \underline{y}) \text{ do}$ 
         $\quad \quad /* \text{ If there were no collider at } \underline{a}, \text{ then there would be info}$ 
         $\quad \quad \text{transmission between } \underline{x} \text{ and } \underline{y} */$ 
         $\quad \quad UE \leftarrow UE - \{\underline{x} - \underline{a}, \underline{a} - \underline{y}\}$ 
         $\quad \quad E \leftarrow E \cup \{\underline{x} \rightarrow \underline{a}, \underline{y} \rightarrow \underline{a}\}$ 
// Orienting edges.
 $again \leftarrow True$ 
 $size \leftarrow |UE|$ 
 $\text{while } again \text{ do}$ 
     $\quad \text{for all } \underline{x} - \underline{y} \in UE \text{ do}$ 
         $\quad \quad \text{if } \underline{x} \rightarrow \underline{y} \in E, \underline{y} - \underline{z} \in UE, \underline{x} - \underline{z} \notin UE, \nexists \underline{w} \ni \underline{w} \rightarrow \underline{y} \in E \text{ then}$ 
             $\quad \quad \quad // \text{ to avoid introducing new v structure}$ 
             $\quad \quad \quad UE \leftarrow UE - \{\underline{y} - \underline{z}\}$ 
             $\quad \quad \quad E \leftarrow E \cup \{\underline{y} \rightarrow \underline{z}\}$ 
         $\quad \quad \text{if } \underline{x} \rightarrow \underline{y} \in E \text{ and there is directed path from } \underline{x} \text{ to } \underline{y} \text{ in } E \text{ then}$ 
             $\quad \quad \quad // \text{ to avoid introducing cycles}$ 
             $\quad \quad \quad UE \leftarrow UE - \{\underline{x} - \underline{y}\}$ 
             $\quad \quad \quad E \leftarrow E \cup \{\underline{x} \rightarrow \underline{y}\}$ 
 $\quad newsize \leftarrow |UE|$ 
 $\quad \text{if } size == newsize \text{ then}$ 
 $\quad \quad | again \leftarrow False$ 
 $\quad \text{else}$ 
 $\quad \quad | size \leftarrow newsize$ 


---



326



return  $G = (V, E, UE)$


```

Chapter 55

Support Vector Machines And Kernel Method

This chapter is based on Refs.[69]. [91] and [70].

The Support Vector Machines (SVM) method was first invented with a linear kernel, but was later generalized to arbitrary kernels. We will use the terms SVM method and Kernel Method indistinguishably.

The SVM method is a fairly general method for calculating, via supervised learning, a *binary* classifier. The SVM method finds a continuous surface that separates a space into two disjoint parts.

Let $\Sigma = [0, 1, 2, \dots, nsam - 1]$ be a list of individuals (samples) in a population. In this chapter, we will use the notation $A^\sigma = A[\sigma]$ and $\vec{A} = [A^\sigma : \sigma \in \Sigma]$ for a list (vector, 1-D array) indexed by Σ . We will refer to $DS = (\vec{x}, \vec{y})$ where $x^\sigma \in S_{\underline{x}}$, $y^\sigma \in \{-1, 1\}$, as a dataset. Let $x^\sigma = (x_0^\sigma, x_1^\sigma, \dots, x_{nf-1}^\sigma) \in S_{\underline{x}_0} \times S_{\underline{x}_1} \times \dots \times S_{\underline{x}_{nf-1}} = S_{\underline{x}}$. When $x_j^\sigma \in \mathbb{R}$ for all j , we will take $x^\sigma \in \mathbb{R}^{nf}$ to be a column vector. x^σ is the feature vector for individual σ , and its components x_i^σ for $i = 0, 1, \dots, nf - 1$ are the features. $y^\sigma \in \{-1, 1\}$ is the binary class to which x^σ belongs.

Let $\hat{y}(x^{\sigma_0}) \in \{-1, 1\}$ be an estimate of $y^{\sigma_0} \in \{-1, 1\}$. The **SVM classifier** is defined as

$$\hat{y}(x^{\sigma_0}) = \text{sign}(Y(x^{\sigma_0})) \quad (55.1)$$

where¹

$$Y(x^{\sigma_0}) = \sum_{\sigma} \alpha^\sigma y^\sigma K(x^\sigma, x^{\sigma_0}). \quad (55.2)$$

The **binary weight coefficients** $\alpha^\sigma \in \{0, 1\}$ for all $\sigma \in \Sigma$ are found by training, via an algorithm to be described below.

The function $K : S_{\underline{x}} \times S_{\underline{x}} \rightarrow \mathbb{R}$ is called the **Kernel or Similarity function**. We will assume that $K(x^\sigma, x^{\sigma_0})$ grows bigger when its two arguments x^σ and x^{σ_0}

¹Define $\text{sign}(0) = 1$.

become more “similar”. We will also assume that $K(x^\sigma, x^{\sigma_0})$ is symmetric in its two arguments.

55.1 Learning Algorithm for SVM Classifier

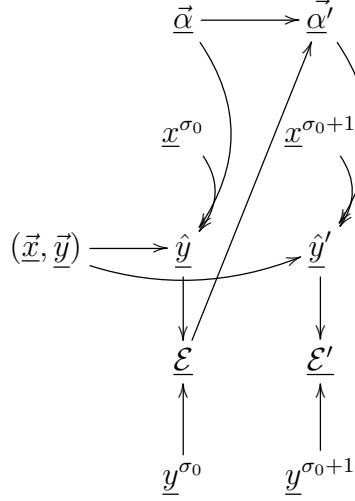


Figure 55.1: Time slice σ_0 of dynamic bnet for learning binary weights $\vec{\alpha}$ of SVM classifier.

Given a kernel function K and a dataset (\vec{x}, \vec{y}) , the SVM classifier is fully specified except for its binary weights $\vec{\alpha}$. Those weights can be learned via the algorithm represented as a causal diagram in Fig.55.1. That figure shows one time slice of a dynamic bnet. The TPMs, printed in blue, of the nodes of the bnet Fig.55.1, are as follows:

$$P(\hat{y}|\vec{\alpha}, (\vec{x}, \vec{y}), x^{\sigma_0}) = \mathbb{1}(\hat{y} = \text{ given by Eq.(55.1).}) \quad (55.3)$$

$$P(\mathcal{E}|\hat{y}, y^{\sigma_0}) = \mathbb{1}(\mathcal{E} = \mathbb{1}(\hat{y} \neq y^{\sigma_0})) \quad (55.4)$$

The first (but not the second, third , etc.) $\vec{\alpha}$ node of Fig.55.1 is a root node. The TPM for that root node should set all components of $\vec{\alpha}$ to zero:

$$P(\vec{\alpha}) = \prod_{\sigma} \mathbb{1}(\alpha^\sigma = 0) . \quad (55.5)$$

After that initialization,

$$P(\vec{\alpha}' | \vec{\alpha}, \mathcal{E}) = \mathbb{1}((\alpha')^\sigma = \alpha^\sigma + \mathcal{E}) \prod_{\sigma \neq \sigma_0} \mathbb{1}((\alpha')^\sigma = \alpha^\sigma) \quad (55.6)$$

Why this learning algorithm works.

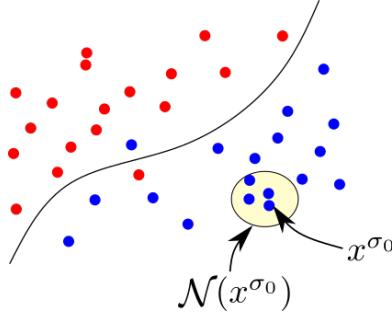


Figure 55.2: Define the neighborhood of x^{σ_0} by $\mathcal{N}(x^{\sigma_0}) = \{x^\sigma : |K(x^\sigma, x^{\sigma_0})| < \epsilon\}$ for some $\epsilon > 0$.

$K(x^\sigma, x^{\sigma_0})$ sets to zero any contribution to $Y(x^{\sigma_0})$ from points x^σ outside the neighborhood $\mathcal{N}(x^{\sigma_0})$ of x^{σ_0} . (See Fig.55.2). If $\hat{y}(x^{\sigma_0}) = y^{\sigma_0}$, keep $\alpha^{\sigma_0} = 0$ because the neighbors of x^{σ_0} are giving the correct $\hat{y}(x^{\sigma_0})$ when they are polled and the majority wins. If, on the other hand, $\hat{y}(x^{\sigma_0}) \neq y^{\sigma_0}$, then switch α^{σ_0} from 0 to 1, which means x^{σ_0} gets to vote by adding y^{σ_0} to $Y(x^{\sigma_0})$. So we start off with all $\alpha^\sigma = 0$ and we end with most of them still zero except for a select few. If we were to set all α^σ equal to one, we would get overfitting and a very jagged separation between the two classes. The fact that we end with only a select few α^σ equal to 1, and the rest equal to 0, helps make the demarcation between the two classes less jagged.

55.2 Linear (dot-product) Kernel

So far, we have discussed the SVM method for an arbitrary kernel. This section is devoted to the **Linear (aka dot-product) Kernel**. Said kernel is defined as

$$K(x^\sigma, x^{\sigma_0}) = (x^\sigma)^T x^{\sigma_0} . \quad (55.7)$$

For this kernel, Eq.(55.2) specializes to

$$Y(x^{\sigma_0}) = \sum_{\sigma} \alpha^\sigma y^\sigma K(x^\sigma, x^{\sigma_0}) + b \quad (55.8)$$

$$= w^T x^{\sigma_0} + b \quad (55.9)$$

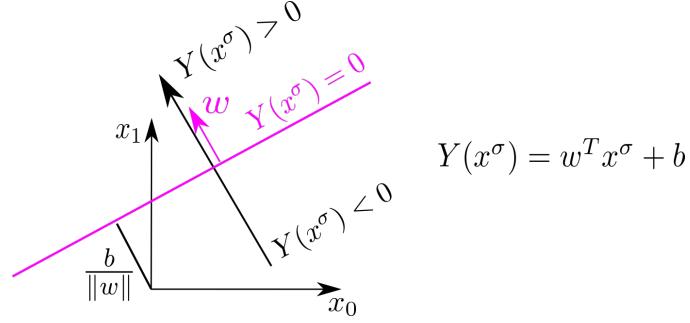


Figure 55.3: Graph of line $Y(x^\sigma) = 0$ splits plane into regions with $Y < 0$, $Y = 0$ and $Y > 0$.

where

$$w = \sum_{\sigma} \alpha^\sigma y^\sigma x^\sigma. \quad (55.10)$$

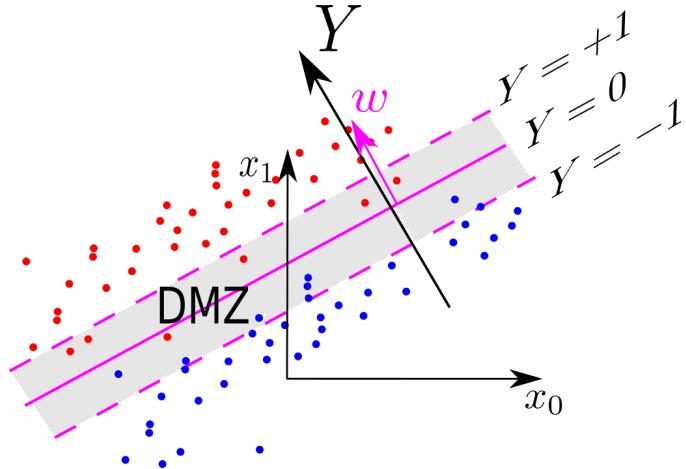


Figure 55.4: We refer to the gray shaded region with $-1 < Y < 1$, where $Y = w^T x + b$, as the DMZ.

We started this chapter by pulling the SVM classifier out of a hat. We did give reasons why it works, but we did not derive it from a more general minimization principle. Such a derivation is possible, at least in the linear kernel case, and we give it next.

Consider the following 3 lines:

$$w^T x^\sigma + b = +A \quad (55.11)$$

$$w^T x^\sigma + b = 0 \quad (55.12)$$

$$w^T x^\sigma + b = -A \quad (55.13)$$

where $w, x^\sigma \in \mathbb{R}^{nf}$, and $b, A \in \mathbb{R}$. We can rescale the vector w and scalar b so as to get rid of the A . This rescaling does not affect the graphs (i.e., x loci) of these 3 lines. Now we have:

$$w^T x^\sigma + b = +1 \quad (55.14)$$

$$w^T x^\sigma + b = 0 \quad (55.15)$$

$$w^T x^\sigma + b = -1 \quad (55.16)$$

If Y stands for

$$Y = w^T x^\sigma + b, \quad (55.17)$$

then we define the **DMZ (demilitarized zone)** to be the region

$$DMZ = \{x^\sigma : |Y(x^\sigma)| < 1\}. \quad (55.18)$$

The lines $Y = \pm 1$ will be called the **borders (aka margins)** of the DMZ, and line $Y = 0$ will be called **line of demarcation** of the DMZ. The DMZ is illustrated in Fig.55.4.

Let D_{DMZ} be the **DMZ width** (i.e., the distance from one border of the DMZ to the other.) Suppose $X_+, X_- \in \mathbb{R}^{nf}$ are two **support vectors** (i.e. vectors on opposite borders with $|X_+ - X_-| = D_{DMZ}$). Then

$$w^T X_+ + b = 1 \quad (55.19)$$

$$w^T X_- + b = -1 \quad (55.20)$$

so

$$D_{DMZ} = \frac{2}{|w|}. \quad (55.21)$$

For any $a \in \mathbb{R}$, let the **positive a_+ and negative a_- parts of a** be given by

$$a = a_+ \mathbf{1}(a > 0) + a_- \mathbf{1}(a \leq 0). \quad (55.22)$$

An **error in $Y(x^\sigma)$** occurs iff $y^\sigma Y(x^\sigma) < 0$. Define the **Cost of erring inside the DMZ** to be

$$CE_{DMZ}(x^\sigma, y^\sigma) = (1 - y^\sigma Y(x^\sigma))_+ = \begin{cases} 1 - y^\sigma Y(x^\sigma) & \text{if } x^\sigma \in DMZ \\ 0 & \text{otherwise} \end{cases} \quad (55.23)$$

Note that the line of demarcation should have the lowest CE_{DMZ} of any line in x^σ space because most of the points on its DMZ have $y^\sigma Y(x^\sigma) > 0$. So to find that line of demarcation, we want to minimize CE_{DMZ} with respect to D_{DMZ} .

However, note also that as D_{DMZ} goes to zero, fewer and fewer data points x^σ fall inside the DMZ. In the limit $D_{DMZ} = 0$, no points do. Since CE_{DMZ} depends on those points and only on those points, $CE_{DMZ} \rightarrow 0$ as $D_{DMZ} \rightarrow 0$.

$\frac{1}{D_{DMZ}}$ is a monotonically decreasing and CE_{DMZ} is a monotonically increasing function of D_{DMZ} . Hence if we add them together, their sum will have a minimum as a function of D_{DMZ} . Define a Lagrangian \mathcal{L} to be the sum of those 2 contributions. $\frac{1}{D_{DMZ}}$ penalizes DMZ's for being too narrow, and CE_{DMZ} penalizes DMZ's for being too wide.

$$\mathcal{L} = \frac{1}{D_{DMZ}} + \sum_{\sigma} CE_{DMZ}(x^\sigma, y^\sigma) \quad (55.24)$$

$$= \frac{1}{2}|w|^2 + \sum_{\sigma} (1 - y^\sigma Y(x^\sigma))_+ \quad (55.25)$$

Setting the variation $\delta\mathcal{L}$ to zero, we get

$$0 = \delta\mathcal{L} = \delta w_i \left\{ w_i - \sum_{\sigma} \mathbb{1}(y^\sigma Y(x^\sigma) < 1) y^\sigma x_i^\sigma \right\} \quad (55.26)$$

so

$$w = \sum_{\sigma} \underbrace{\mathbb{1}(y^\sigma Y(x^\sigma) < 1)}_{\alpha^\sigma} y^\sigma x^\sigma. \quad (55.27)$$

55.3 Alternatives to Linear Kernel

Sometimes it is convenient to switch the dot-product kernel given above by a different kernel. Other popular kernels are

- **Gaussian Kernel. Radial function Basis (RFB) Kernel.** Uses for K a radial function; i.e., a function that only depends on the magnitude (radius, Euclidean distance, L^2 norm) of a vector.

$$K(x^\sigma, x^{\sigma_0}) = e^{-\gamma|x^\sigma - x^{\sigma_0}|^2} \quad (55.28)$$

for some free parameter $\gamma > 0$.

- **Inhomogeneous Polynomial Kernel**

$$K(x^\sigma, x^{\sigma_0}) = [(x^\sigma)^T x^{\sigma_0} + 1]^d \quad (55.29)$$

for some positive integer d .

- **Homogeneous Polynomial Kernel**

$$K(x^\sigma, x^{\sigma_0}) = [(x^\sigma)^T x^{\sigma_0}]^d \quad (55.30)$$

for some positive integer d .

- **”Kernel trick” Kernel.** Consider a map $\Phi : \mathbb{R}^{nf} \rightarrow \mathbb{R}^N$. Usually $N > nf$. Φ can be a rectangular matrix $T \in \mathbb{R}^{N \times nf}$ so that $\Phi(x^\sigma) = Tx^\sigma \in \mathbb{R}^N$. Let

$$K(x^\sigma, x^{\sigma_0}) = [\Phi(x^\sigma)]^T \Phi(x^{\sigma_0}) \quad (55.31)$$

Although the constant surfaces of this kernel are hyperplanes in \mathbb{R}^N , its constant surfaces in \mathbb{R}^{nf} can be curved and even closed compact surfaces (e.g. spheres).

55.4 Random Forest and Kernel Method

Chapter 56

Synthetic Controls

This chapter is based on Refs.[5] and [3].

This chapter assumes that the reader has read Chapter 13 on the Difference-in-Differences (DID) method.

The Synthetic Controls (SC) method is a simple enhancement of the DID method. SC enhances DID in two simple yet powerful ways:

1. **Better time resolution.** DID considers just 2 time-snapshots (i.e., a time-series with only 2 times) whereas SC considers arbitrarily many time-snapshots (i.e., a time-series with more than 2 times).
2. **Weighted average of controls.** DID divides the population of individuals into just 2 kinds: the treated and the untreated (aka controls). SC divides the total population into treated and controls just like DID does, but it goes further and divides the control population into multiple subpopulations, and calculates a weighted average, called a “synthetic control”, of those subpopulations. The weights of the synthetic control are chosen so that it mimics as closely as possible the behavior of the treated population for all times measured before the treatment was applied.

Let us describe these two enhancements more precisely.

- **timing:** Let t_k for $k = 0, 1, \dots, n_{pre} - 1$ be the pre-treatment times at which a measurement occurs. Let t_k for $k = n_{pre}, n_{pre} + 1, \dots, n_{times} - 1$ be the post-treatment times at which a measurement occurs. Note that $n_{pre} + n_{post} = n_{times}$. Note that $t_* = t_{n_{pre}+1}$ is the first measurement time after the treatment is applied, t_0 is the first measurement time, and $t_{fin} = t_{n_{times}-1}$ is the last one.
- **subpopulations:** Let $S_1 = \{\sigma_1\}$ be the set of treated units (just one). Let $S_0 = \{\sigma : \sigma \neq \sigma_1\}$ be the set of untreated units (i.e., controls). Let $nsam$ = number of all units σ , $n_1 = |S_1| = 1$, and $n_0 = |S_0| = nsam - 1$.
- **weights:**

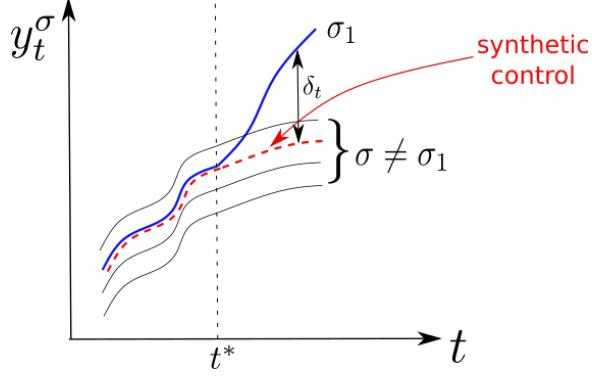


Figure 56.1: Pictorial representation of the Synthetic Controls (SC) method. The outcome y of the synthetic control unit is colored red and that of the treated unit is colored blue. They roughly agree for $t < t_*$.

We want to define a time-independent weight w^σ for each unit σ in such a way that the output y_t^σ for the synthetic control unit behaves like the output for the treated unit σ_1 for $t < t_*$.

Let

$$w^{\sigma_1} = 0 \quad (56.1)$$

and

$$w^{n_0} = \{w^\sigma\}_{\sigma \neq \sigma_1}. \quad (56.2)$$

Define a cost function \mathcal{C} :

$$\mathcal{C}(w^{n_0}) = \sum_{t < t_*} \left(y_t^{\sigma_1} - \sum_{\sigma \neq \sigma_1} w^\sigma y_t^\sigma \right)^2 \quad (56.3)$$

Then calculate w^{n_0} by minimizing the cost function, subject to the constraint that w^{n_0} be a probability distribution:

$$w^{n_0} = \operatorname{argmin}_{W^{n_0}} \left\{ \mathcal{C}(W^{n_0}) : W^\sigma \geq 0, \sum_{\sigma \neq \sigma_1} W^\sigma = 1 \right\}. \quad (56.4)$$

Now that we have defined a weight w^σ for every unit σ , we can define

$$y_t^\xi = \begin{cases} y_t^{\sigma_1} & \text{if } \xi = 1 \\ \sum_{\sigma \neq \sigma_1} w^\sigma y_t^\sigma & \text{if } \xi = 0 \end{cases} \quad (56.5)$$

and

$$\delta_t = y_t^{\xi=1} - y_t^{\xi=0} \quad (56.6)$$

δ_t is illustrated in Fig.56.1. It measures the time dependent gap (causal effect) between the outcome (i.e., y) of the treated unit σ_1 and the outcome of the synthetic control unit.

56.1 A bnet G_t with weighted treatment outcomes

Our next goal is to analyze the SC method using the formalism of PO theory. To attain that goal, we will first define in this section a bnet G_t . The bnet G_t in this chapter differs in two important respects from the bnet G_t defined in Chapter 13 on the DID method. First, the G_t in the DID chapter is defined for only 2 measurement times whereas the G_t in this chapter is defined for more than 2 measurement times. Second, the G_t in this chapter defines \underline{y}_t for $d = 0$ as a weighted average over control subpopulations.

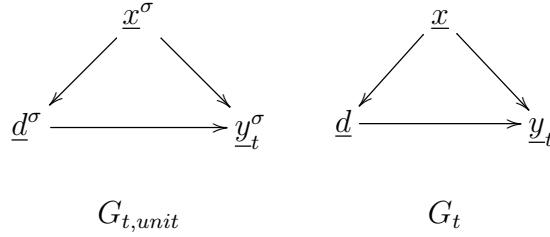


Figure 56.2: $t \in \{t_0, t_1, \dots, t_{fin}\}$. The bnet G_t is defined by counting units σ for the bnet $G_{t,unit}$.

Suppose $t \in \{t_0, t_1, \dots, t_{fin}\}$. For $\xi \in \{0, 1\}$, define the following unit counts:

$$N_{d,y_t,x}^{\xi} = \sum_{\sigma \in S_{\xi}} \mathbb{1}(d = d^{\sigma}, y_t = y_t^{\sigma}, x = x^{\sigma}) . \quad (56.7)$$

Define also

$$N_{d,y_t,x} = \sum_{\xi \in \{0,1\}} \mathbb{1}(\xi, d) N_{d,y_t,x}^{\xi} . \quad (56.8)$$

Henceforth, sums over the subscripts of $N_{d,y_t,x}$ will be indicated by a dot. For example, $N_{\cdot,y_t,x} = \sum_d N_{d,y_t,x}$.

The TPMs, printed in blue, for the bnet G_t shown in Fig.56.2, are as follows.

$$P_{\underline{x}}(x) = \frac{N_{\cdot,\cdot,x}}{N_{\cdot,\cdot,\cdot}} \quad (56.9)$$

$$P_{\underline{d}|\underline{x}}(d|x) = \frac{N_{d,\cdot,x}}{N_{\cdot,\cdot,x}} \quad (56.10)$$

$$P_{\underline{y}_t|\underline{d},\underline{x}}(y_t|d, x) = \frac{N_{d,y_t,x}}{N_{d,\cdot,x}} \quad (56.11)$$

56.2 PO analysis

In this section, we show how to analyze the SC method using the formalism of PO theory.

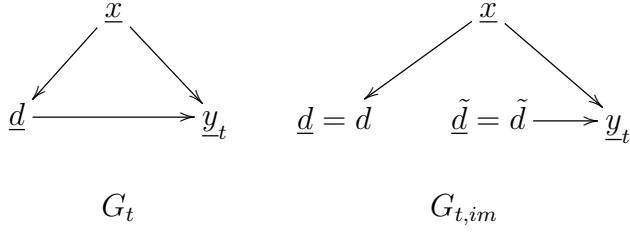


Figure 56.3: $t \in \{t_0, t_1, \dots, t_{fin}\}$. Bnet $G_{t,im} = \kappa_{\underline{d} \rightarrow \underline{y}_t}(\tilde{d})G_t$ is obtained by applying the imagine operator to arrow $\underline{d} \rightarrow \underline{y}_t$ of bnet G_t .

As usual for PO theory, we will consider expected values of y_t^σ :

$$E_{\sigma|\tilde{d},x}[y_t^\sigma(d)] = E_{y_t|\tilde{d},x}[y_t(d)] = \mathcal{Y}_{d|\tilde{d},x}(t) \quad (56.12)$$

To calculate these expected values, we need a “model” with probability distributions. In this case, the needed model and probability distributions are provided by the bnets depicted in Fig.56.3. The TPMs, printed in blue, for the bnet $G_{t,im}$ in Fig.56.3, are as follows. Note that the TPMs for the bnet $G_{t,im}$ are defined in terms of the TPMs for the bnet G_t .

$$P(x) = P_{\underline{x}}(x) \quad (56.13)$$

$$P(d|x) = P_{d|\underline{x}}(d|x) \quad (56.14)$$

$$P(y_t|\tilde{d}, x) = P_{\underline{y}_t|\underline{d},\underline{x}}(y_t|\tilde{d}, x) \quad (56.15)$$

$$P((\tilde{d})') = \delta((\tilde{d})', \tilde{d}) \quad (56.16)$$

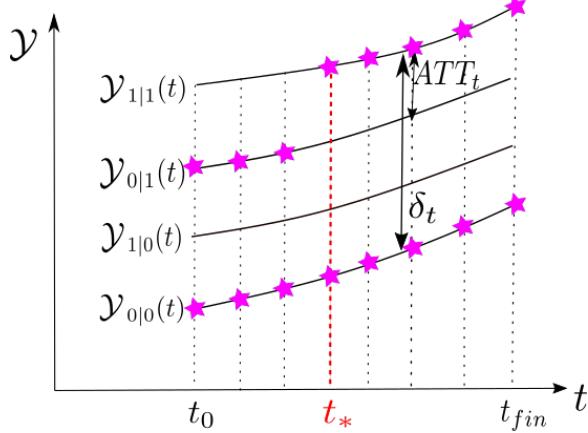


Figure 56.4: Four different time-dependent expected values $\mathcal{Y}_{d|\tilde{d}}(t)$ of y_t^σ for bnet $G_{t,im}$. The $2 * ntimes$ magenta stars represents the $2 * ntimes$ SC measurements.

Henceforth, for simplicity, we will omit the confounder state x from the indices of \mathcal{Y} ; i.e., we will write $\mathcal{Y}_{d|\tilde{d}}(t)$ instead of $\mathcal{Y}_{d|\tilde{d},x}(t)$. The fact that we will not explicitly mention x does not mean that it doesn't exist or that it doesn't affect our analysis. If there are confounders, they cannot be neglected. As discussed in Chapter 43 under the subject of strata-matching in PO, one must condition \mathcal{Y} on a single x stratum and, later on, one must average over all the possible x strata.

Let $\mathcal{MY}_{d|\tilde{d}}(t)$ denote the **measured** $\mathcal{Y}_{d|\tilde{d}}(t)$. We define this quantity as

$$\mathcal{MY}_{d|\tilde{d}}(t) = \mathcal{Y}_{d|\tilde{d}}(t) \left[\mathbb{1}(d=0, t < t_*) + \mathbb{1}(d=\tilde{d}, t \geq t_*) \right] \quad (56.17)$$

Now we claim that the SC δ_t calculated in the previous section can be expressed in PO formalism as follows:

$$\delta_t = \mathcal{Y}_{1|1}(t) - \mathcal{Y}_{0|0}(t) = SDO_t \quad (56.18)$$

for $t \geq t_*$. Fig.56.4 depicts the four functions $\mathcal{Y}_{d|\tilde{d}}(t)$ for t in the interval $[t_0, t_{fin}]$ and for $d, \tilde{d} \in \{0, 1\}$. The \mathcal{Y} coordinates of the $2 * ntimes$ magenta stars in Fig.56.4 can be calculated using bnet G_t . Note that in Fig.56.4, we display a large gap between the curves $\mathcal{Y}_{0|\tilde{d}}(t)$ for $\tilde{d} \in \{0, 1\}$. In reality, $P(y_t|\tilde{d}, x)$ has been constructed so as to make that gap as small as possible. Thus, to a good(?) approximation,

$$\delta_t \approx ATT_t \quad (56.19)$$

Hence, unlike in the DID method, in the SC method, to a good(?) approximation, we don't have to worry about parallel trends.

Chapter 57

Transformer Networks

This chapter is based on Refs.[13] and [92].

Transformer Networks (TN) have been taking the field of Natural Language Processing (NLP) by storm in recent years. They were introduced in 2017 and already are the basis of BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), two TN libraries that have been trained with huge databases such as all of English Wikipedia (2,500M words).

Recurrent Neural Nets (RNNs) are discussed in Chapter 46. TNs are quickly displacing RNNs, an older method, in NLP. TNs are better than RNNs for doing NLP in several important ways. Whereas RNNs analyze the tokens (words) of a sentence sequentially (like a Kalman Filter), TNs analyze them in parallel, and thus are more amenable to parallel computing. Also, because RNNs analyze the words of a sentence sequentially, they tend to give more importance to the end of a sentence than to its beginning. That's because RNNs start forgetting the beginning of a sentence by the time they reach its end, like a patient with Alzheimer's. TNs do not suffer from this malady.

Dynamic bnets are discussed in Chapter 17. In Chapter 46, we showed that RNNs are dynamic bnets. The goal of this chapter is to define TNs, and to show that they too are dynamic bnets.

Let

\mathcal{S} be the set of words in a sentence,

$h_i^t \in \mathbb{R}^{nh}$ be an nh dimensional column vector for word $i \in \mathcal{S}$ at time t .

$Q^t, K^t, V^t \in \mathbb{R}^{nh \times nh}$ be the **prior-attention matrices for time slice t** .

These matrices are learned by training the net. The letters Q, K, V stand for Query, Key and Value, respectively.

Fig.57.1 represents a TN of a 3-word sentence as a dynamic bnet. The TPMs, printed in blue, for the nodes of the bnet Fig.57.1, are as follows:

$$P(V^t) = \delta(V^t, V_0^t) \quad (57.1a)$$

$$\begin{array}{ccc}
\underline{V}^t & \underline{Q}^t & \underline{K}^t \\
\downarrow & \downarrow & \downarrow \\
\{\underline{v}_i^t\}_{i=0,1,2} & \{\underline{q}_i^t\}_{i=0,1,2} & \{\underline{k}_i^t\}_{i=0,1,2}
\end{array}$$

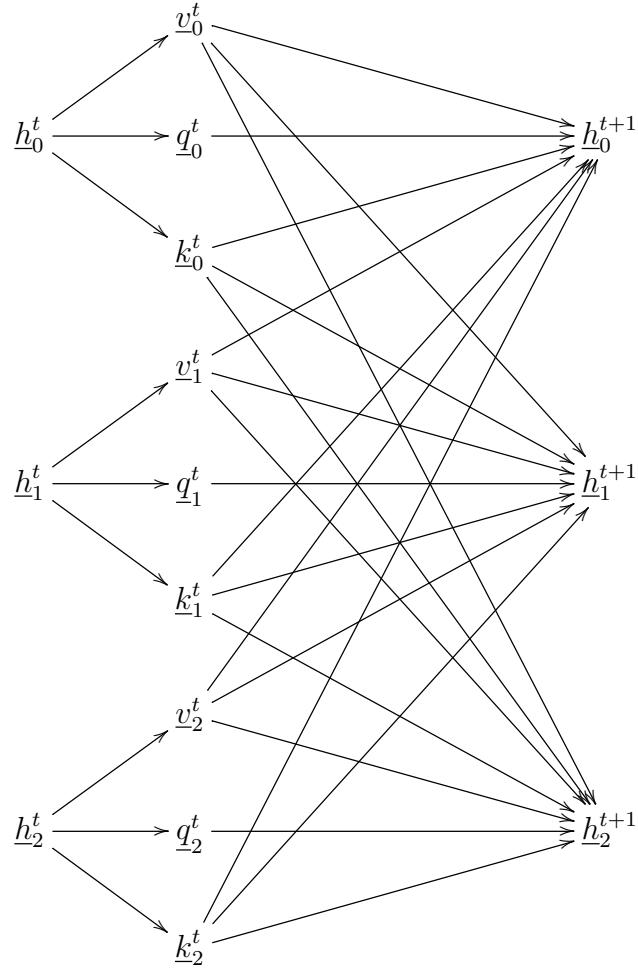


Figure 57.1: Time slice t of dynamic bnet for a transformer network (TN) of a 3-word sentence. The following arrows were not drawn explicitly for clarity: arrows pointing from node \underline{V}^t to nodes \underline{v}_i^t for $i = 0, 1, 2$, from node \underline{Q}^t to nodes \underline{q}_i^t for $i = 0, 1, 2$, and from node \underline{K}^t to nodes \underline{k}_i^t for all i . Note that k_i^t for all i points to \underline{h}_j^{t+1} for all j . Likewise, \underline{v}_i^t for all i points to \underline{h}_j^{t+1} for all j .

$$P(Q^t) = \delta(Q^t, Q_0^t) \quad (57.1b)$$

$$P(K^t) = \delta(K^t, K_0^t) \quad (57.1c)$$

$$P(v_i^t | V^t, h_i^t) = \mathbb{1}(v^t = V^t h_i^t) \quad (57.2)$$

$$P(q_i^t | Q^t, h_i^t) = \mathbb{1}(q^t = Q^t h_i^t) \quad (57.3)$$

$$P(k_i^t | K^t, h_i^t) = \mathbb{1}(k^t = K^t h_i^t) \quad (57.4)$$

$$P(h_i^{t+1} | v_i^t, q_i^t, k_i^t) = \mathbb{1}(h_i^{t+1} = \sum_{j \in \mathcal{S}} v_j^t \underbrace{\frac{e^{(q_i^t)^T k_j^t}}{\sum_{j' \in \mathcal{S}} e^{(q_i^t)^T k_{j'}^t}}}_{\substack{w_{j|i} = [\text{softmax}((q_i^t)^T k_j^t)]_j \\ E_{j|i}[v_j^t] = \text{Attention}}} \quad (57.5)$$

In Eqs.57.1, the 3 root nodes $\underline{V}^t, \underline{Q}^t, \underline{K}^t$ have delta function priors. To improve stability of the dynamic bnet, it is more common instead to use for these priors, multiple delta functions (aka **multi-heads**) labeled $c = 0, 1, \dots, nc - 1$, as follows.

$$P(V^t | c) = \delta(V^t, V_c^t) \quad (57.6a)$$

$$P(Q^t | c) = \delta(Q^t, Q_c^t) \quad (57.6b)$$

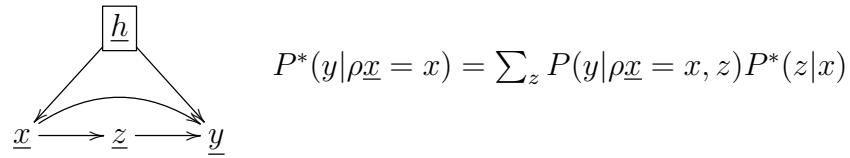
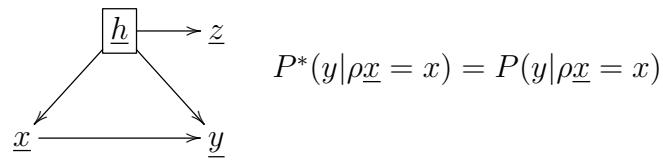
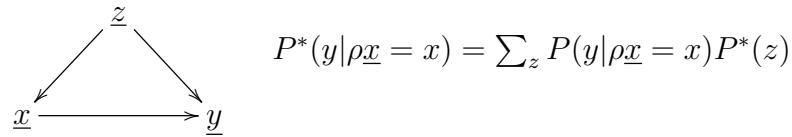
$$P(K^t | c) = \delta(K^t, K_c^t) \quad (57.6c)$$

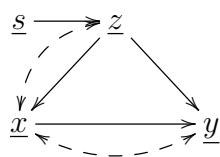
for $c = 0, 1, \dots, nc - 1$. Even better, one can go fully Bayesian, and make these 3 prior distributions arbitrary categorical distributions to be determined by net training, instead of mere delta functions.

Chapter 58

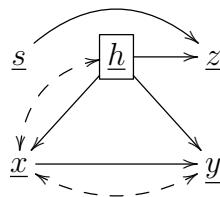
Transportability: COMING SOON

Ref.[29]

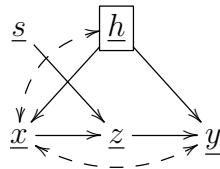




$$P(y|\rho \underline{x} = x, s) = \sum_z P(y|\rho \underline{x} = x)P(z|s)$$



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



$$P(y|\rho \underline{x} = x, s) = \sum_z P(y|\rho \underline{x} = x, z)P(z|x, s)$$

Chapter 59

Turbo Codes

This chapter is based on Ref.[18].

In this chapter, vectors with n components will be indicated by an n superscript. For example, $a^n = (a_0, a_1, \dots, a_{n-1})$.

Consider an n -letter message $u^n = (u_0, u_1, \dots, u_{n-1})$, where for all i , $u_i \in \mathcal{A}$ is an element of an alphabet \mathcal{A} , and where for all i , the 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.59.1, or by its abridged version Fig.59.2. 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).

Define

$$x = (u^n, e_1, e_2) \quad (59.1)$$

and

$$\tilde{x} = (\tilde{u}^n, \tilde{e}_1, \tilde{e}_2) . \quad (59.2)$$

Fig.59.1 implies that

$$P(x, \tilde{x}) = P(\tilde{u}^n | u^n) \left[\prod_{r=1,2} P(\tilde{e}_r | e_r) P(e_r | u^n) \right] P(u^n) . \quad (59.3)$$

Because the u^n are i.i.d.,

$$P(u^n) = \prod_i P(u_i) . \quad (59.4)$$

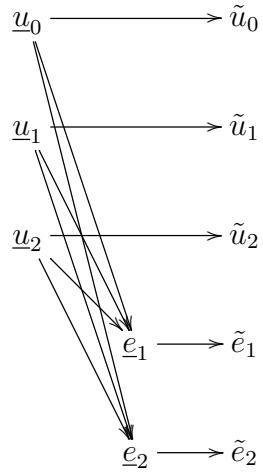


Figure 59.1: Turbo coding Bnet representing a message being encoded two different ways and then the original message and the 2 encodings pass through a memoryless channel.

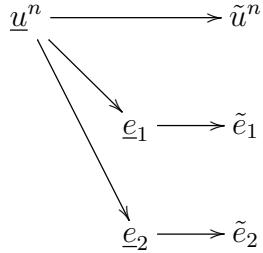


Figure 59.2: Abridged version of Fig.59.1.

Because the channel is memoryless,

$$P(\tilde{u}^n | u^n) = \prod_i P(\tilde{u}_i | u_i). \quad (59.5)$$

Because the encoding is deterministic, we must have for $r = 1, 2$

$$P(e_r | u^n) = \delta(e_r, e_r(u^n)). \quad (59.6)$$

Define the belief functions

$$BEL_i = BEL_i(\underline{u}_i = a) = P(\underline{u}_i = a | \tilde{x}). \quad (59.7)$$

The best estimate of u_j given all visible evidence \tilde{x} is

$$\hat{u}_i = \operatorname{argmax}_{u_i} BEL_i(u_i) . \quad (59.8)$$

Define the probability functions

$$\pi_i = \pi_i(u_i) = P(u_i) , \quad (59.9)$$

and the likelihood functions

$$\lambda_i = \lambda_i(u_i) = P(\tilde{u}_i|u_i) . \quad (59.10)$$

For $r = 1, 2$, define the Kernel functions

$$K_r = K_r(u^n) = P(\tilde{e}_r|e_r = e_r(u^n)) . \quad (59.11)$$

In this book, $\mathcal{N}(!a)$ denotes a normalization constant that does not depend on a . Define

$$\mathcal{N}_i = \mathcal{N}(!u_i) . \quad (59.12)$$

Claim 31

$$BEL_i = \mathcal{N}_i \lambda_i \pi_i \mathcal{T}_i^{K_1 K_2} \left[\prod_{j \neq i} \lambda_j \pi_j \right] , \quad (59.13)$$

where $\mathcal{T}_i^K(\cdot)$ with $K = K_1 K_2$ is an operator (transform) that acts on functions of u^n :

$$\mathcal{T}_i^K(\cdot) = \sum_{u^n} \delta(u_i, a) K(u^n)(\cdot) . \quad (59.14)$$

proof:

$$\begin{aligned} P(\underline{u}_i = a | \tilde{x}) &= \\ &= \sum_x \delta(u_i, a) P(x | \tilde{x}) \end{aligned} \quad (59.15)$$

$$= \sum_x \delta(u_i, a) \frac{P(\tilde{x}|x)P(x)}{P(\tilde{x})} \quad (59.16)$$

$$= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(\tilde{x}|x)P(x) \quad (59.17)$$

$$= \mathcal{N}(!a) \sum_x \delta(u_i, a) P(u^n) \left[\prod_{r=1,2} P(\tilde{e}_r|e_r) \delta(e_r, e_r(u^n)) \right] \prod_j P(\tilde{u}_j|u_j) \quad (59.18)$$

$$= \mathcal{N}(!a) \lambda_i(a) \pi_i(a) R , \quad (59.19)$$

where

$$R = \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} P(\tilde{e}_r | e_r(u^n)) \right] \prod_{j \neq i} P(\tilde{u}_j | u_j) P(u_j) \quad (59.20)$$

$$= \sum_{u^n} \delta(u_i, a) \left[\prod_{r=1,2} K_r(u^n) \right] \prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j) \quad (59.21)$$

$$= \mathcal{T}_i^{K_1 K_2} \left[\prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j) \right]. \quad (59.22)$$

Hence

$$BEL_i(a) = \mathcal{N}(!a) \lambda_i(a) \pi_i(a) \mathcal{T}_i^{K_1 K_2} \left[\prod_{j \neq i} \lambda_j(u_j) \pi_j(u_j) \right]. \quad (59.23)$$

QED

59.1 Decoding Algorithm

The Turbo algorithm for decoding the encode message is as follows. For $m = 0$, let

$$\pi_j^{(0)}(u_j) = \frac{1}{n_{\underline{u}_j}}. \quad (59.24)$$

Then for $m = 1, 2, \dots$, let

$$\pi_i^{(m)} = \mathcal{N}_i \mathcal{T}_i^{K_m \% 2} \left[\prod_{j \neq i} \lambda_j \pi_j^{(m-1)} \right], \quad (59.25)$$

where $m \% 2 = 1$ if m is odd and $m \% 2 = 2$ if m is even. Furthermore, for $m > 0$, let

$$BEL_i^{(m)} = \mathcal{N}_i \lambda_i \pi_i^{(m-1)} \pi_i^{(m)} \quad (59.26)$$

$$= \mathcal{N}_i \lambda_i \pi_i^{(m-1)} \mathcal{T}_i^{K_m \% 2} \left[\prod_{j \neq i} \lambda_j \pi_j^{(m-1)} \right]. \quad (59.27)$$

As $m \rightarrow \infty$, $BEL_i^{(m)}$ given by Eq.(59.27) is expected to converge to the the exact BEL_i given by Eq.(59.13).

Turbo decoding can be represented by the bnets Figs.59.3 and 59.4.

The node TPMs, printed in blue, for Fig.59.3, are given by:

$$P(d_i^{(m)} = a | \tilde{u}^n, \tilde{e}_{m \% 2}) = BEL_i^{(m)}(a). \quad (59.28)$$

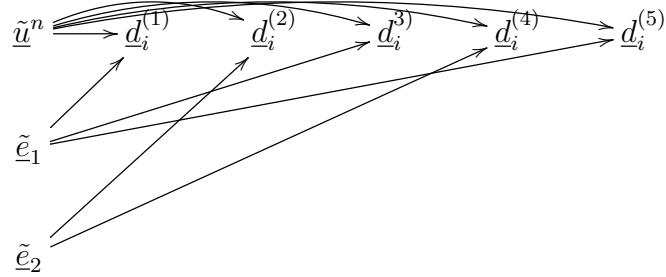


Figure 59.3: Bnet describing Turbo code generation of $BEL_i^{(m)}(a)$ for $m = 1, 2, \dots$

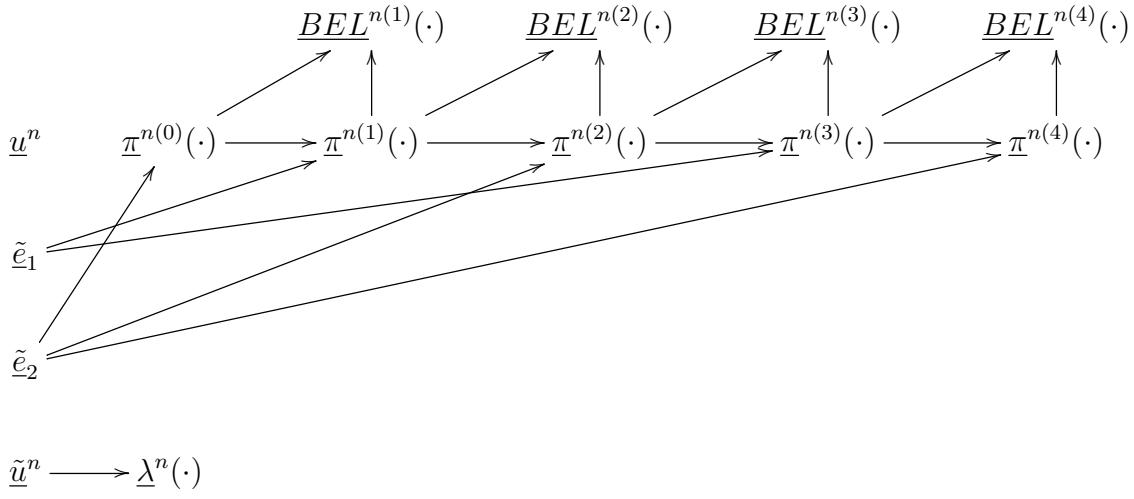


Figure 59.4: Bnet describing Turbo code generation of $BEL^{n(m)}(\cdot)$ and $\pi^{n(m)}(\cdot)$ for $m = 0, 1, 2, \dots$. The following arrows were not drawn for clarity: Arrows pointing from node $\lambda^n(\cdot)$ to nodes $\underline{\pi}^{n(m)}(\cdot)$ and $\underline{BEL}^{n(m)}(\cdot)$ for $m = 0, 1, 2, \dots$

The TPMs, printed in blue, for Fig.59.4, are given by:

$$P((\lambda^n)'(\cdot) | \tilde{\underline{u}}^n) = \delta((\lambda^n)'(\cdot), \lambda^n(\cdot)) \quad (59.29)$$

$$P(\pi^{n(m)}(\cdot) | \lambda^n(\cdot), \pi^{n(m-1)}(\cdot), \tilde{\underline{e}}_{m \% 2}) = \prod_i \prod_{u_i} \delta(\pi_i^{(m)}(u_i), \mathcal{N}_i \mathcal{T}_i^{K_m \% 2} [\prod_{j \neq i} \lambda_j \pi_j^{(m-1)}]) \quad (59.30)$$

$$P(BEl^{n(m)}(\cdot) | \lambda^n(\cdot), \pi^{n(m)}(\cdot), \pi^{n(m-1)}(\cdot)) = \prod_i \prod_{u_i} \delta(BEL_i(u_i), \mathcal{N}_i \lambda_i \pi_i^{(m-1)} \pi_i^{(m)})$$
(59.31)

59.2 Message Passing Interpretation of Decoding Algorithm

Ref.[18] 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 35.

Chapter 60

Uplift Modelling

This chapter is based on many references, including Ref.[10, 6, 93, 33].

Uphill Modelling (UP) deals with the application of Rubin’s Theory of Potential Outcomes (PO) to advertisement and marketing.

PO, which is discussed in Chapter 43, is a subset of Pearl’s Causal Inference. Besides UP, other applications of PO theory that are discussed in this book are: Regression Discontinuity (Chapter 47), Difference-in-Differences (Chapter 13) and Synthetic Controls (Chapter 56).

In UP, each **participant person** is interrogated at two well anticipated, fairly closely spaced times t_0 and t_1 (as opposed to Difference-in-Differences (DID), where t_0 and t_1 might be years apart, and long before the DID analysis is attempted.). In between those two times, a treatment which we will refer to as the **UP diagnostic test** is applied. For example, at times t_0 and t_1 , every participant might be asked how important he/she rates climate change on a scale of 1 to 10. In between times t_0 and t_1 , every participant might be sent a brochure on climate change. In UP, as in all other PO applications, each **sample** σ is in the treated or control groups, but not both. But in UP, the same participant can be in both the treated and control groups. If so, that participant is considered two different samples σ ; for example, $\sigma = \text{treatedBob}, \text{controlBob}$. In UP, the samples are aware of which of those groups they are in, so they are not “treatment blind”. As explained in Chapter 43, strata matching is only valid if the samples of the test are treatment blind. Therefore, the only treatment effect that makes sense for UP is SDO, because it does not require strata matching.

60.1 UP types

Let $y_t^B \in \mathbb{R}$ for $t = t_0, t_1$ be the treatment response at time t for participant B . (We are using here the same notation as in Chapter 43). Call $\delta^B = y_{t_1}^B - y_{t_0}^B$ the **participant uplift** for participant B . As shown in Fig.60.1, UP classifies participants into 4 **UP-types**: Persuadables, SureThings, LostCauses, and SleepyDogs. The UP-type of a participant depends on the changes that are induced on that participant by an

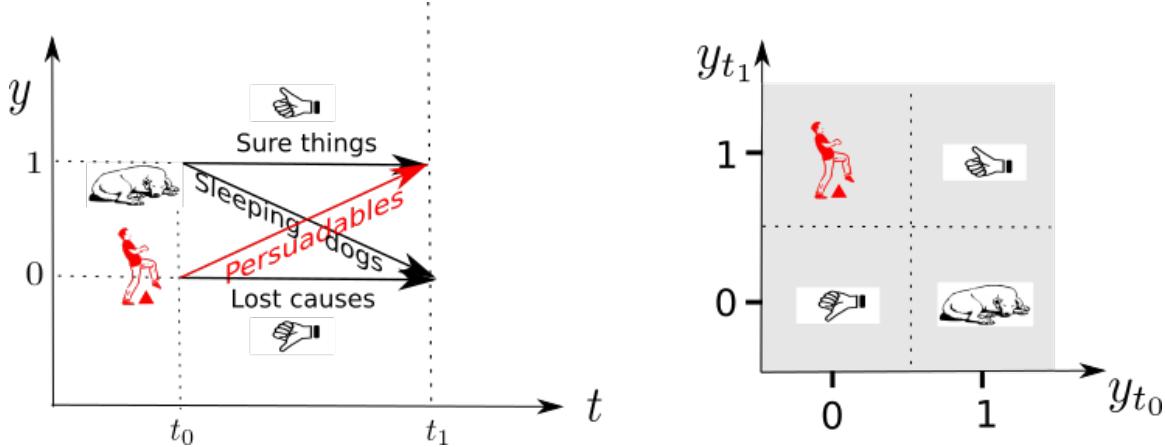


Figure 60.1: UP diagnostic test can be used to classify all participants of the population into 4 UP-types. This figure assumes $y \in \{0, 1\}$. More generally, $y \in \mathbb{R}$. t represents time. $t = t_0$ corresponds to $d = 0 = \text{untreated}$, and $t = t_1$ corresponds to $d = 1 = \text{treated}$.

UP-diagnostic-test.

- For a **Persuadable** participant, $\delta^B > 0$.
- For a **SleepyDogs** participant, $\delta^B < 0$.
- For a **SureThings** participant, $\delta^B \approx 0$ and $y_{t_0}^B$ is high.
- For a **LostCauses** participant, $\delta^B \approx 0$ and $y_{t_0}^B$ is low.

Suppose B belongs to stratum A_x . What is commonly called the **uplift** is the **stratum-uplift** $\delta_x = SDO$. Strata can also be classified into the 4 UP-types, depending on the sign and size of their δ_x . A participant may not be typical for his stratum and may have different **participant and stratum UP-types**. For example, he may have positive participant uplift and therefore have a Persuadable participant UP-type, but his stratum-uplift might be negative, so he has the SleepyDogs stratum UP-type.

Advertisers are very interested in finding the Persuadable strata in a population so as to focus their resources on them. For example, UP was used very successfully during the Obama presidential campaigns. Team Obama conducted UP-diagnostic tests much like the climate change one described earlier. This allowed them to identify voters who might be sitting on the fence on whether to vote for Obama or not. Then Team Obama spent the lion share of resources on those fence-sitters.

60.2 Some Relevant Technical Facts from Chapter 43

Some relevant technical facts about SDO that were proven in Chapter 43 are

- $SDO = 0$ is the hypothesis tested by a Randomized Clinical Trial (RTC).
- Using $\underline{y} = \underline{y}(\tilde{d})$ and Fig.43.7, we get

$$SDO = \sum_y y \left[P_{\underline{y}(1)|\tilde{d},\underline{x}}(y|1,x) - P_{\underline{y}(0)|\tilde{d},\underline{x}}(y|0,x) \right] \quad (60.1)$$

$$= \sum_y y \left[P_{\underline{y}|\tilde{d},\underline{x}}(y|1,x) - P_{\underline{y}|\tilde{d},\underline{x}}(y|0,x) \right]. \quad (60.2)$$

If $\underline{y} \in \{0, 1\}$, then

$$\underbrace{\delta_x}_{\widehat{SDO}} = \underbrace{P_{\underline{y}|\tilde{d},\underline{x}}(1|1,x)}_{Y_x^1} - \underbrace{P_{\underline{y}|\tilde{d},\underline{x}}(1|0,x)}_{Y_x^0}. \quad (60.3)$$

- Recall that in Chapter 43, we used $A_{d,x} = \{\sigma : \tilde{d}^\sigma = d, x^\sigma = x\}$, $A_x = A_{0,x} \cup A_{1,x}$ and $A = \cup_x A_x$; also $N_{d,x} = |A_{d,x}|$, $N_x = |A_x|$ and $N = |A|$. From Eq.(43.87), we get

$$\widehat{\delta_x} = \underbrace{\frac{1}{N_{1,x}} \sum_{\sigma \in A_{1,x}} y^\sigma}_{Y_x^1} - \underbrace{\frac{1}{N_{0,x}} \sum_{\sigma \in A_{0,x}} y^\sigma}_{Y_x^0}. \quad (60.4)$$

60.3 UP Analysis

The input to UP is a PO dataset $DS = \{(\sigma, d^\sigma, x^\sigma, y^\sigma) : \sigma = 0, 1, 2, \dots, nsam - 1\}$. where $d^\sigma \in \{0, 1\}$, $x^\sigma \in S_{\underline{x}}$, $y^\sigma \in \mathbb{R}$. A participant B is assigned two different σ if he/she belongs to both the treated and control groups. We will assume $S_{\underline{x}}$ is a finite set. In general, $x = (x_0, x_1, \dots, x_{n-1})$ is an n dimensional vector of features x_i . If any of the x_i is a priori continuous, we will assume it has been binned into a finite number of bins.

Starting with DS , UP performs the following steps. Fig.60.2 is a pictorial representation of the quantities that are calculated during these steps.

1. Find A_x for each observed $x \in S_{\underline{x}}$. Set $A_x = \emptyset$ for unobserved $x \in S_{\underline{x}}$.

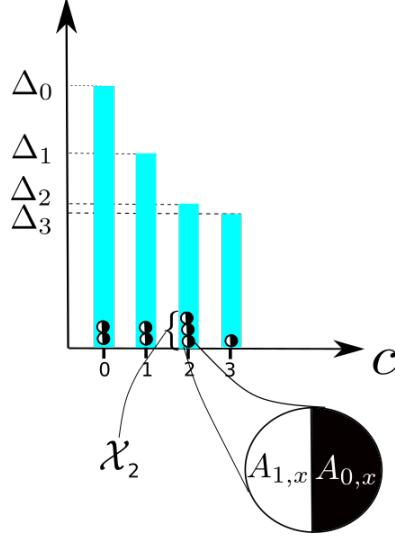


Figure 60.2: Pictorial representation of the sequence $\{(\mathcal{X}_c, \Delta_c)\}_{c=0,1,\dots,nc-1}$.

2. Calculate δ_x for each $x \in S_{\underline{x}}$. Set $\delta_x = 0$ if $A_x = \emptyset$.
3. Calculate the set

$$\{\Delta_c\}_{c=0,1,\dots,nc-1} = \{\delta_x : x \in S_{\underline{x}}\} \quad (60.5)$$

of distinct uplifts δ_x . The class labels c should be assigned so that the sequence of Δ_c is monotonic and non-increasing; i.e.,

$$\Delta_0 \geq \Delta_1 \geq \cdots \geq \Delta_{nc-1} . \quad (60.6)$$

Now calculate

$$\mathcal{X}_c = \{x : \delta_x = \Delta_c\} \quad (60.7)$$

for each c . By the end of this step, we will have calculated $\{(\mathcal{X}_c, \Delta_c)\}_{c=0,1,\dots,nc-1}$. We will refer to the \mathcal{X}_c as **strata-bins**. Note that

$$\Delta_c = \frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} \delta_x \quad (60.8)$$

$$= \underbrace{\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} Y_x^1}_{Y_c^1} - \underbrace{\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} Y_x^0}_{Y_c^0} . \quad (60.9)$$

4. For each c , calculate

$$\Sigma_{d,c} = \bigcup_{x \in \mathcal{X}_c} A_{d,x} \quad (60.10)$$

for $d \in \{0, 1\}$ and

$$\Sigma_c = \Sigma_{0,c} \cup \Sigma_{1,c} . \quad (60.11)$$

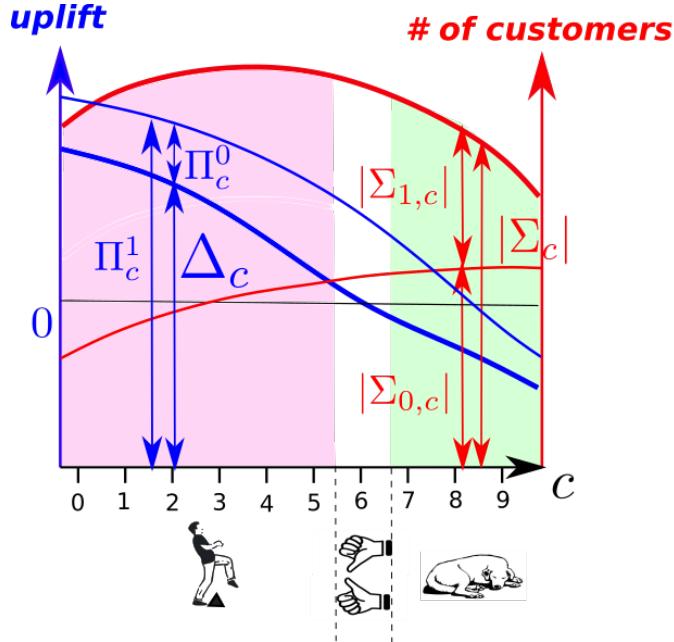


Figure 60.3: Plot of UP results. Alternative to Qini curves.

Fig.60.3 is a way of plotting the results of UP in an intuitive way that even a business type can understand. UP software often plots something called a Qini curve, but I find Qini curves opaque, confusingly defined in the literature, unnecessary and not very well motivated. So I don't use them.

60.4 UP Decision Trees

In this section, we will describe how to build UP decision trees (UP dtrees), and explain why they are needed for UP.

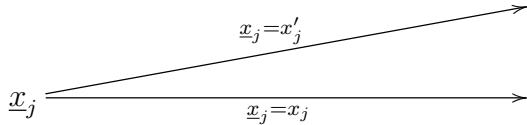
Generic dtrees are described in Chapter 12. This section complements rather than replaces that chapter so the reader is advised to read that chapter first.

Ref.[33] is an excellent paper on the use of dtrees in UP.

The analysis described previously in Section 60.3, although theoretically correct, will work very poorly in practice. The strata-bins of Section 60.3 correspond to the classification classes of a dtree. But strata-bins are very specific so they severely overfit the data. Although dtrees can also suffer from overfitting, there are known methods of preventing or mitigating overfitting in dtrees.

There are also tasks that dtrees can do well and the methods explained so far cannot do well. For example, suppose we have a classless dataset $DS^- = \{(\sigma, x^\sigma) : \sigma \in \Sigma^-\}$ and we want to predict the class c^σ and uplift Δ_{c^σ} for each of these individuals $\sigma \in \Sigma^-$. A dtree can easily do that. The alternative is to use the classy dataset $DS = \{(\sigma, x^\sigma, c^\sigma) : \sigma \in \Sigma\}$ to prepare a dictionary that orders the elements of $S_{\underline{x}}$ and gives a class c and an uplift value Δ_c for each feature vector $x \in S_{\underline{x}}$. But such a dictionary overfits and says nothing for feature vectors x that do not show up in the classy dataset DS ; i.e., the dictionary doesn't guess (interpolate). Dtrees, on the other hand, do guess.

So, without further ado, let us describe how to modify the results of Chapter 12 on generic dtrees to the case of UP dtrees. The main difference, as we will explain in detail next, is that the Information Gain metric used for generic dtrees needs to be replaced by another metric.



$$\{N_j^d(c, x_j)\}_{c \in S_{\underline{c}}, x_j \in S_{\underline{x}_j}}$$

$$\sum_{c \in S_{\underline{c}}} N_j^d(c, x_j) = N_j^d(x_j)$$

$$\sum_{x_j \in S_{\underline{x}_j}} N_j^d(c, x_j) = N_j^d(c) \quad \sum_{c \in S_{\underline{c}}} N_j^d(c) = \sum_{x_j \in S_{\underline{x}_j}} N_j^d(x_j) = N_j^d$$

Figure 60.4: Fig.12.4 with d dependence added. $d \in \{0, 1\}$ is the treatment dose.

Fig.60.4 was obtained from Fig.12.4 by adding d dependence. $d \in \{0, 1\}$ is the treatment dose. Note that in UP, we build a dtree in which every node carries a double ($d = 0, 1$) TPV. This is in contrast to the generic dtrees built in Chapter 12, in which each node carries a single TPV. $N_j^d(c, x_j)$ is the number of individuals σ in

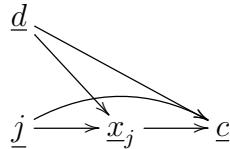


Figure 60.5: Bnet derived from population numbers in Fig.60.4

the population that reaches node \underline{x}_j with $d \in \{0, 1\}$, belonging to class $c \in S_c$ and having $\underline{x}_j = x_j$. From these population numbers, we can define the bnet in Fig.60.5. The TPMs, printed in blue, for the (non-root) nodes of this bnet, are as follows

$$P(c|x_j, j, d) = \frac{N_j^d(c, x_j)}{N_j^d(x_j)} \quad (60.12)$$

$$P(x_j|j, d) = \frac{N^d(x_j)}{N_j^d} \quad (60.13)$$

In Chapter 12, we used Information Gain (a mutual information) as the SAM (Separation Ability Measure) in SL (Structure Learning) of dtrees (Decision Trees). Information Gain is a bad SAM for SL of UP dtrees, because it knows nothing about $d = 0, 1$ and the double TPVs of nodes in UP dtrees. For UP dtrees, we need a SAM specifically designed to separate $d = 0, 1$, and generate classes that are uplift bins (i.e., uplift intervals).

Ref.[33] proposes and studies the following 3 SAMs for doing SL of UP dtrees.

1. SAM_DD (DD=Delta Delta)

For $d \in \{0, 1\}$ and $c, c' \in S_c$, define the increments

$$\partial_d f(d) = f(1) - f(0) \quad (60.14)$$

and

$$\partial_{c',c} f(c) = f(c') - f(c) . \quad (60.15)$$

Let

$$\Delta_{c|j} = P(c|j, 1) - P(c|j, 0) \quad (60.16)$$

$$= \partial_d P(c|j, d) \quad (60.17)$$

$$SAM_DD_j = \max_{c, c'} |\partial_{c',c} \partial_d P(c|j, d)| \quad (60.18)$$

$$= \max_{c, c'} |\partial_{c',c} \Delta_{c|j}| \quad (60.19)$$

2. SAM_KL (KL=Kullback Liebler)

$$SAM_KL_j = \left[\sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) D_{KL}(P_{\underline{c}|x_j,j,1} \| P_{\underline{c}|x_j,j,0}) \right] - D_{KL}(P_{\underline{c}|j,1} \| P_{\underline{c}|j,0}) \quad (60.20)$$

$$= \left[\sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) \sum_{c \in S_c} P(c|x_j, j, 1) \ln \frac{P(c|x_j, j, 1)}{P(c|x_j, j, 0)} \right] - \sum_{c \in S_c} P(c|j, 1) \ln \frac{P(c|j, 1)}{P(c|j, 0)} \quad (60.21)$$

SAM_KL_j can be negative.

3. SAM_E (E=Euclidean)

SAM_E_j is defined the same way as SAM_KL_j except with the KL divergence $D_{KL}(P \| Q)$ in SAM_KL replaced by the Euclidean distance squared.

$$D(P, Q) = \sum_x (P(x) - Q(x))^2 \quad (60.22)$$

The intuitive reason for using these quantities as SAMs is that they maximize the change in uplift between successive tree levels, so that the uplift increases as quickly as possible as we descend down the UP tree. In the case of generic dtrees for which we use Information Gain as SAM, we are maximizing the correlation between classes and nodes as we descend down the tree. These two goals are related. In fact, in the limit where the number of control individuals becomes zero, SAM_KL_j and IG_j become the same, as will be shown later.

Next we show that SAM_KL_j satisfies the following 3 axioms¹

Claim 32 .

1. SAM_KL_j is minimum iff $P(c|x_j, j, 0) = P(c|x_j, j, 1)$ for all c and x_j .
2. If $P(c|j, d) = P(c|d)$ for all c, d , then $SAM_KL_j = 0$.
3. Suppose $N_r^0 = 0$ for all nodes $r \in J_0$ (i.e., no control population) and we use the Laplace Correction when warranted. Then

$$SAM_KL_j = H(\underline{c} : \underline{x}_j | j, 1) \quad (60.23)$$

$$= IG_j \quad \text{for treated population.} \quad (60.24)$$

¹ We won't show it here, but according to Ref.[33], SAM_E_j also satisfies these 3 axioms, but SAM_DD_j satisfies only the first two.

proof:

The proof of items 1 and 2 follow by inspection of Eq.60.21. Item 3 is proven in Claim 33 below.

QED

Let $N_{\underline{c}} = |S_{\underline{c}}|$. Define the uniform probability distribution

$$U_{\underline{c}}(c) = \frac{1}{N_{\underline{c}}} \quad (60.25)$$

for all $c \in S_{\underline{c}}$.

Eq.(60.12) for the TPM of node \underline{c} in the bnet Fig.60.5 can be "Laplace Corrected" as follows so that it is no longer undefined when its denominator vanishes:

$$P(c|j, d) = \begin{cases} \frac{N_j^d(c)}{N_j^d} & \text{if } N_j^d > 0 \\ U_{\underline{c}}(c) & \text{if } N_j^d = 0 \text{ (Laplace Correction)} \end{cases} \quad (60.26)$$

Claim 33 Suppose $N_r^0 = 0$ for all dtree nodes $r \in J_0$ and we use the Laplace Correction when warranted. Then

$$SAM_KL_j = H(\underline{c} : \underline{x}_j | j, 1) . \quad (60.27)$$

proof:

For all nodes $r \in J_0$, we must have

$$P_{\underline{c}|r,0} = U_{\underline{c}} \quad (60.28)$$

so

$$D_{KL}(P_{\underline{c}|r,1} \| P_{\underline{c}|r,0}) = D_{KL}(P_{\underline{c}|r,1} \| U_{\underline{c}}) \quad (60.29)$$

$$= \ln(N_{\underline{c}}) - H(\underline{c}|r, 1) . \quad (60.30)$$

For all $x_j \in S_{\underline{x}_j}$, we must also have

$$N_j = N_j^1, N(x_j) = N^1(x_j) \quad (60.31)$$

so

$$P(x_j | j) = P(x_j | j, 1) . \quad (60.32)$$

Now using Eqs.(60.30) and (60.32), we get

$$SAM_KL_j = - \left[\sum_{x_j \in S_{\underline{x}_j}} P(x_j|j) H(\underline{c}|x_j, j, 1) \right] + H(\underline{c}|j, 1) \quad (60.33)$$

$$= - \left[\sum_{x_j \in S_{\underline{x}_j}} P(x_j|j, 1) H(\underline{c}|x_j, j, 1) \right] + H(\underline{c}|j, 1) \quad (60.34)$$

$$= -H(\underline{c}|\underline{x}_j, j, 1) + H(\underline{c}|j, 1) \quad (60.35)$$

$$= H(\underline{c} : \underline{x}_j | j, 1) \quad (60.36)$$

QED

60.4.1 Appendix, connection between Δ_c and $\Delta_{c|j}$

Recall Eq.60.9:

$$\Delta_c = \underbrace{\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} Y_x^1}_{Y_c^1} - \underbrace{\frac{1}{|\mathcal{X}_c|} \sum_{x \in \mathcal{X}_c} Y_x^0}_{Y_c^0} \quad (60.37)$$

$$= \partial_d Y_c^d. \quad (60.38)$$

Compare that to Eq.(60.17):

$$\Delta_{c|j} = P(c|j, 1) - P(c|j, 0) \quad (60.39)$$

$$= \partial_d P(c|j, d) \quad (60.40)$$

What is the connection between these 2 deltas, Δ_c and $\Delta_{c|j}$? Are they equal?

First off, notice that $\Delta_{c|j}$ is defined for all nodes j of the dtree. Let $j(c)$ be the leaf node for which $\Delta_c \approx \Delta_{c|j(c)}$. Assume $y^\sigma \in \{0, 1\}$. Then

$$P(c|j = j(c), d) = \frac{N_{j(c)}^d(c)}{N_{j(c)}^d} \approx Y_c^d \quad (60.41)$$

So the two deltas are indeed approximately equal when $y^\sigma \in \{0, 1\}$ and $j = j(c)$.

Chapter 61

Variational Bayesian Approximation

For more info and references about this topic, see Ref.[94].

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 $\underline{h} \in S_h$ and $\underline{q} \in S_h$. Suppose $\underline{\vec{x}} \in S_{\vec{x}}^{nsam}$ is a vector of $nsam$ samples and the samples $\underline{x}[\sigma] \in S_x$ are i.i.d.. The VBA is simply an approximation $P_{\underline{q}|\vec{x}}$ to $P_{\underline{h}|\vec{x}}$:

$$P_{\underline{h}|\vec{x}}(h|\vec{x}) \approx P_{\underline{q}|\vec{x}}(h|\vec{x}) \quad (61.1)$$

obtained by minimizing the Kullback-Liebler divergence $D_{KL}(P_{\underline{q}|\vec{x}} \| P_{\underline{h}|\vec{x}})$ over all $P_{\underline{q}|\vec{x}}$. The minimization is usually subject to some constraints on the admissible forms of $P_{\underline{q}|\vec{x}}$.

$D_{KL}(Q \| P) \neq D_{KL}(P \| Q)$; i.e., D_{KL} is not symmetric. So why do we use $D_{KL}(P_{\underline{q}|\vec{x}} \| P_{\underline{h}|\vec{x}})$ instead of $D_{KL}(P_{\underline{h}|\vec{x}} \| P_{\underline{q}|\vec{x}})$? Because $D_{KL}(P_{\underline{h}|\vec{x}} \| P_{\underline{q}|\vec{x}})$ requires knowledge of $P_{\underline{h}|\vec{x}}$, but calculating $P_{\underline{h}|\vec{x}}$ is what we are trying to do in the first place.

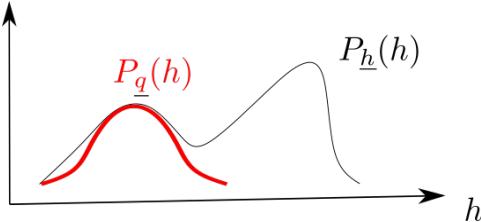


Figure 61.1: If $P_{\underline{q}}(h)$ is Gaussian shaped and $P_{\underline{h}}(h)$ has multiple bumps (modes) then $D_{KL}(P_{\underline{q}} \| P_{\underline{h}})$ is minimized when $P_{\underline{q}}$ fits one of the modes of $P_{\underline{h}}$. That is because $D_{KL}(P_{\underline{q}} \| P_{\underline{h}}) = \sum_h P_{\underline{q}}(h) \ln \frac{P_{\underline{q}}(h)}{P_{\underline{h}}(h)}$ 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.61.1 for some intuition on what minimizing $D_{KL}(P_{\underline{q}|\vec{x}} \| P_{\underline{h}|\vec{x}})$ means.

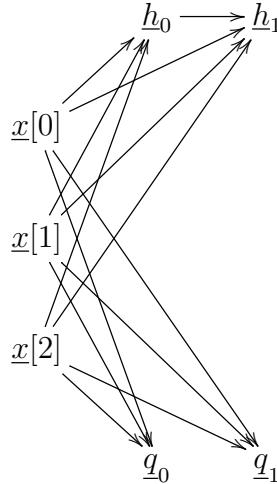


Figure 61.2: \underline{q} and \underline{h} have $nh = 2$ mirroring components and those of \underline{q} are independent at fixed \underline{x} .

Suppose $\underline{h} = (h_0, h_1, \dots, h_{nh-1})$ and $\underline{q} = (q_0, q_1, \dots, q_{nh-1})$ where $h_i \in S_{h_i}$ and $q_i \in S_{h_i}$ for all i . We say \underline{q} and \underline{h} have nh mirroring components and those of \underline{q} are independent at fixed \underline{x} if

$$P_{\underline{q}|\underline{x}}(\underline{h}|\vec{x}) = \prod_i P_{q_i|\underline{x}}(h_i|\vec{x}). \quad (61.2)$$

The bnet Fig.61.2 describes the scenario that we have in mind: The samples $x[\sigma]$ are i.i.d.. Each component h_i of \underline{h} has a mirroring component q_i in \underline{q} . The components of \underline{h} are correlated whereas those of \underline{q} are independent at fixed \underline{x} .

Claim 34 *If \underline{q} and \underline{h} have nh mirroring components and those of \underline{q} are independent at fixed \underline{x} and $D_{KL}(P_{\underline{q}|\underline{x}} \| P_{\underline{h}|\underline{x}})$ is minimum over all $P_{\underline{q}|\underline{x}}$, then*

$$P_{q_i|\underline{x}}(q_i|\vec{x}) = \mathcal{N}(!q_i) e^{E_{(q_j)_{j \neq i}} [\ln P_{\underline{h}|\underline{x}}(\underline{h}=q|\vec{x})]} \quad (61.3)$$

$$= \mathcal{N}(!q_i) e^{E_{(q_j)_{j \neq i}} [\ln P_{\underline{h}, \underline{x}}(\underline{h}=q, \vec{x})]} \quad (61.4)$$

for all i .

proof:

Since all quantities in Eq.(61.3) are conditioned on \vec{x} , let us omit all mention of \vec{x} in this proof.

Let

$$\mathcal{L} = \mathcal{L}_0 + \mathcal{L}_1 \quad (61.5)$$

where

$$\mathcal{L}_0 = D_{KL}(P_{\underline{q}} \parallel P_{\underline{h}}) \quad (61.6)$$

$$= \sum_h P_{\underline{q}}(h) \ln \frac{P_{\underline{q}}(h)}{P_{\underline{h}}(h)} \quad (61.7)$$

$$= \sum_h P_{\underline{q}}(h) \ln P_{\underline{q}}(h) - \sum_h P_{\underline{q}}(h) \ln P_{\underline{h}}(h) \quad (61.8)$$

$$= \sum_i \sum_{h_i} P_{\underline{q}_i}(h_i) \ln P_{\underline{q}_i}(h_i) - \sum_h P_{\underline{q}}(h) \ln P_{\underline{h}}(h) \quad (61.9)$$

and

$$\mathcal{L}_1 = \sum_i \lambda_i \left[\sum_{h_i} P_{\underline{q}_i}(h_i) - 1 \right]. \quad (61.10)$$

Then

$$\delta \mathcal{L} = \sum_i \sum_{h_i} \delta P_{\underline{q}_i}(h_i) \left[\ln P_{\underline{q}_i}(h_i) + 1 + \lambda_i - \frac{1}{nh} \sum_{(h_j)_{j \neq i}} \prod_{(h_j)_{j \neq i}} \{P_{\underline{q}_j}(h_j)\} \ln P_{\underline{h}}(h) \right]. \quad (61.11)$$

Hence,

$$P_{\underline{q}_i}(h_i) = \mathcal{N}(!h_i) e^{\sum_{(h_j)_{j \neq i}} \left\{ \prod_{(h_j)_{j \neq i}} P_{\underline{q}_j}(h_j) \right\} \ln P_{\underline{h}}(h)}. \quad (61.12)$$

QED

Note that Eq.(61.3) yields a system of nh nonlinear equations in nh unknowns $(P_{\underline{q}_i|\vec{x}})_{i=0,1,\dots,nh-1}$. This system is usually solved recursively.

61.1 Free Energy $\mathcal{F}(\vec{x})$

To simplify the notation below, let us introduce the following abbreviations:

$$P(h|\vec{x}) = P_{\underline{h}|\vec{x}}(h|\vec{x}) \quad (61.13)$$

$$P(h, \vec{x}) = P_{\underline{h}, \vec{x}}(h, \vec{x}) \quad (61.14)$$

$$P(\vec{x}) = P_{\vec{x}}(\vec{x}) \quad (61.15)$$

Note that

$$D_{KL}(P_{\underline{q}|\vec{x}} \parallel P_{\underline{h}|\vec{x}}) = \sum_h P_{\underline{q}|\vec{x}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\vec{x}}(h|\vec{x})}{P(h|\vec{x})} \quad (61.16)$$

$$= \sum_h P_{\underline{q}|\vec{x}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\vec{x}}(h|\vec{x})}{P(h, \vec{x})} + \ln P(\vec{x}) \quad (61.17)$$

$$= \mathcal{F}(\vec{x}) + \ln P(\vec{x}) \quad (61.18)$$

Hence, the **Free energy** $\mathcal{F}(\vec{x})$ is defined as

$$\mathcal{F}(\vec{x}) = \sum_h P_{\underline{q}|\vec{x}}(h|\vec{x}) \ln \frac{P_{\underline{q}|\vec{x}}(h|\vec{x})}{P(h, \vec{x})} \quad (61.19)$$

$$= E_{\underline{q}|\vec{x}} \left[\ln \frac{P_{\underline{q}|\vec{x}}(q|\vec{x})}{P_{\underline{h},\vec{x}}(q, \vec{x})} \right]. \quad (61.20)$$

The name free energy is justified because

$$\mathcal{F}(\vec{x}) = \underbrace{- \sum_h P_{\underline{q}|\vec{x}}(h|\vec{x}) \ln P_{\underline{h},\vec{x}}(h, \vec{x})}_{U, \text{ Internal Energy}} + \underbrace{\sum_h P_{\underline{q}|\vec{x}}(h|\vec{x}) \ln P_{\underline{q}|\vec{x}}(h|\vec{x})}_{-S, \text{ minus Entropy}}. \quad (61.21)$$

It is also common to define a quantity called “ELBO” to be the negative of the free energy.

$$ELBO(\vec{x}) = -\mathcal{F}(\vec{x}) \quad (61.22)$$

ELBO stands for “Evidence Lower BOund”. That name is justified because

$$\underbrace{\ln P_{\vec{x}}(\vec{x})}_{\text{evidence} \leq 0} = \underbrace{D_{KL}(P_{\underline{q}|\vec{x}} \parallel P_{\underline{h}|\vec{x}})}_{\geq 0} - |ELBO(\vec{x})|. \quad (61.23)$$

Some properties of \mathcal{F} are:

- \mathcal{F} is non-negative.

$$\underbrace{D_{KL}(P_{\underline{q}|\vec{x}} \parallel P_{\underline{h}|\vec{x}})}_{\geq 0} + \underbrace{\ln \frac{1}{P_{\vec{x}}(\vec{x})}}_{\geq 0} = \mathcal{F}(\vec{x}) \quad (61.24)$$

- KL divergence is min iff \mathcal{F} is min at fixed $P(\vec{x})$.

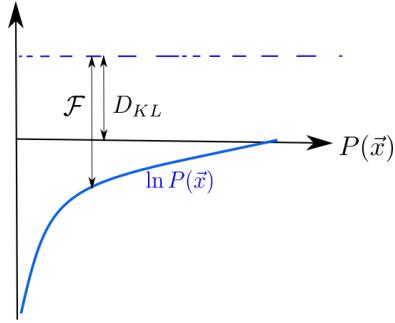


Figure 61.3: $D_{KL} + \ln \frac{1}{P(\vec{x})} = \mathcal{F}$.

During a variation δ that holds $P(\vec{x})$ fixed, the KL divergence and \mathcal{F} change by the same amount:

$$\delta D_{KL}(P_{g|\vec{x}} \parallel P_{h|\vec{x}}) = \delta \mathcal{F}(\vec{x}) \quad (61.25)$$

Chapter 62

Zero Information Transmission (Graphoid Axioms)

This chapter assumes that you have read Chapter 16 on d-separation.

The following quantities play a very prominent role in the d-separation Theorem that we enunciated in Chapter 16.

- 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):

$$H(\underline{y} : \underline{x}^n) = \sum_i H(\underline{y} : \underline{x}_i | \underline{x}_{<i}), \quad (62.1)$$

where $\underline{x}^n = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{n-1})$ and $\underline{x}_{<i} = (\underline{x}_0, \underline{x}_1, \dots, \underline{x}_{i-1})$.

A trivial but very useful consequence of the chain rule for CMI is:

$$\boxed{H(\underline{y} : \underline{x}^n) = 0 \iff H(\underline{y} : \underline{x}_i | \underline{x}_{<i}) = 0 \text{ for all } i}. \quad (62.2)$$

62.1 Consequences of Eq.(62.2)

Table 62.1 gives a set of statements about CMI referred to as the Graphoid Axioms in chapter 1 of Ref.[27]. See Ref.[27] 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.(62.2).

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 62.1: Graphoid Axioms

Claim 35 *Table 62.1 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

$$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{b}|\underline{c}) + H(\underline{a} : \underline{c}) , \quad (62.3)$$

and

$$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{c}|\underline{b}) + H(\underline{a} : \underline{b}) . \quad (62.4)$$

Hence,

$$H(\underline{a} : \underline{b}, \underline{c}) = 0 \quad (62.5)$$

implies

$$H(\underline{a} : \underline{b}|\underline{c}) = H(\underline{a} : \underline{c}) = 0 , \quad (62.6)$$

and

$$H(\underline{a} : \underline{c}|\underline{b}) = H(\underline{a} : \underline{b}) = 0 . \quad (62.7)$$

- **Weak Union**

Already proven in proof of Decomposition.

- **Contraction**

From chain rule for CMI, we have

$$H(\underline{a} : \underline{b}, \underline{c}) = H(\underline{a} : \underline{b} | \underline{c}) + H(\underline{a} : \underline{c}) . \quad (62.8)$$

- **Intersection**

From the chain rule for CMI, we have

$$H(\underline{a} : \underline{b}, \underline{d} | \underline{c}) = H(\underline{a} : \underline{b} | \underline{d}, \underline{c}) + H(\underline{a} : \underline{d} | \underline{c}) , \quad (62.9)$$

and

$$H(\underline{a} : \underline{b}, \underline{d} | \underline{c}) = H(\underline{a} : \underline{d} | \underline{b}, \underline{c}) + H(\underline{a} : \underline{b} | \underline{c}) . \quad (62.10)$$

Thus,

$$H(\underline{a} : \underline{b}, \underline{d} | \underline{c}) = 0 \quad (62.11)$$

implies

$$H(\underline{a} : \underline{b} | \underline{d}, \underline{c}) = H(\underline{a} : \underline{d} | \underline{c}) = 0 , \quad (62.12)$$

and

$$H(\underline{a} : \underline{d} | \underline{b}, \underline{c}) = H(\underline{a} : \underline{b} | \underline{c}) = 0 . \quad (62.13)$$

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] Scott Cunningham. *Causal inference: The mixtape*. Yale University Press, 2021. <https://mixtape.scunning.com/index.html>.
- [4] Robin J. Evans. Graphical methods for inequality constraints in marginalized DAGs. <https://arxiv.org/abs/1209.2978>.
- [5] Matheus Facure Alves. *Causal Inference for The Brave and True*. 2021. <https://matheusfacure.github.io/python-causality-handbook/landing-page.html>.
- [6] George Fei. Modeling uplift directly: Uplift decision tree with kl divergence and euclidean distance as splitting criteria. <https://tinyurl.com/yhnzwj58>.
- [7] 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>.
- [8] Bruno Gonçalves. Model testing and causal search. blog post <https://medium.com/data-for-science/causal-inference-part-vii-model-testing-and-causal-search-536b796f0384>.
- [9] 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>.
- [10] Pierre Gutierrez and Jean-Yves Grardy. Causal inference and uplift modelling: A review of the literature. In *Proceedings of The 3rd International Conference on Predictive Applications and APIs*, pages 1–13, 2017. <http://proceedings.mlr.press/v67/gutierrez17a.html>.
- [11] Christina Heinze-Deml. Causality, spring semester 2019 at ETH Zurich. https://stat.ethz.ch/lectures/ss19/causality.php#course_materials.

- [12] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International journal of approximate reasoning*, 15(3):225–263, 1996. <http://www.artiste.com/Huang-Darwiche1996.pdf>.
- [13] Chaitanya K. Joshi. Transformer (machine learning model). <https://graphdeeplearning.github.io/post/transformers-are-gnns/>.
- [14] 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>.
- [15] Sergey Levine. Course CS 285 at UC Berkeley, Deep reinforcement learning. <http://rail.eecs.berkeley.edu/deeprlcourse/>.
- [16] Dimitris Margaritis. Learning Bayesian network model structure from data (thesis, 2003, Carnegie Mellon Univ). <https://apps.dtic.mil/sti/citations/ADA461103>.
- [17] 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>.
- [18] 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>.
- [19] Richard E Neapolitan. *Learning Bayesian networks*. Pearson Prentice Hall, 2004.
- [20] Andrew Ng. Lecture at deeplearning.ai on recurrent neural networks. <http://www.artiste.com/ng-lec-rnn.pdf>.
- [21] Gregory Nuel. Tutorial on exact belief propagation in Bayesian networks: from messages to algorithms. <https://arxiv.org/abs/1201.4724>.
- [22] Judea Pearl. Linear models: A useful microscope for causal analysis. https://ftp.cs.ucla.edu/pub/stat_ser/r409-corrected-reprint.pdf.
- [23] Judea Pearl. Mediating instrumental variables. https://ftp.cs.ucla.edu/pub/stat_ser/r210.pdf.
- [24] Judea Pearl. On the testability of causal models with latent and instrumental variables. <https://arxiv.org/abs/1302.4976>.

- [25] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. <https://www.aaai.org/Papers/AAAI/1982/AAAI82-032.pdf>, 1982.
- [26] Judea Pearl. *Probabilistic Inference in Intelligent Systems*. Morgan Kaufmann, 1988.
- [27] Judea Pearl. *Causality: Models, Reasoning, and Inference, Second Edition*. Cambridge University Press, 2013.
- [28] Judea Pearl. Causal and counterfactual inference. *The Handbook of Rationality*, pages 1–41, 2019. https://ftp.cs.ucla.edu/pub/stat_ser/r485.pdf.
- [29] Judea Pearl and Elias Bareinboim. Transportability of causal and statistical relations: A formal approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011. <https://ojs.aaai.org/index.php/AAAI/article/view/7861>.
- [30] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.
- [31] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.
- [32] ReliaSoft. System analysis reference. http://reliawiki.org/index.php/System_Analysis_Reference.
- [33] Piotr Rzepakowski and Szymon Jaroszewicz. Decision trees for uplift modeling with single and multiple treatments. *Knowledge and Information Systems*, 32(2):303–327, 2012. <https://link.springer.com/content/pdf/10.1007/s10115-011-0434-0.pdf>.
- [34] Marco Scutari. bnlearn. <https://www.bnlearn.com/>.
- [35] 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>.
- [36] 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>.
- [37] Russell Stuart and Norvig Peter. Artificial intelligence-a modern approach 3rd ed, 2016.

- [38] 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/>.
- [39] theinvestorsbook.com. Pert analysis. <https://theinvestorsbook.com/pert-analysis.html>.
- [40] Robert R. Tucci. Bell's inequalities for Bayesian statisticians. blog post in blog Quantum Bayesian Networks, <https://qbnets.wordpress.com/2008/09/19/bells-inequations-for-bayesian-statistician/>.
- [41] Robert R. Tucci. Goodness of causal fit. <https://github.com/rrtuucci/goodness-c-fit/raw/master/gcf.pdf>.
- [42] Robert R. Tucci. Quantum Fog. <https://github.com/artiste-qb-net/quantum-fog>.
- [43] 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/>.
- [44] Wikipedia. AdaBoost. <https://en.wikipedia.org/wiki/AdaBoost>.
- [45] Wikipedia. Belief propagation. https://en.wikipedia.org/wiki/Belief_propagation.
- [46] Wikipedia. Berkson's paradox. https://en.wikipedia.org/wiki/Berkson%27s_paradox.
- [47] Wikipedia. Beta function. https://en.wikipedia.org/wiki/Beta_function.
- [48] Wikipedia. Binary decision diagram. https://en.wikipedia.org/wiki/Binary_decision_diagram.
- [49] Wikipedia. Boolean algebra. https://en.wikipedia.org/wiki/Boolean_algebra.
- [50] Wikipedia. Bootstrap aggregating. https://en.wikipedia.org/wiki/Bootstrap_aggregating.
- [51] Wikipedia. Categorical distribution. https://en.wikipedia.org/wiki/Categorical_distribution.
- [52] Wikipedia. Chow-Liu tree. https://en.wikipedia.org/wiki/Chow%E2%80%93Liu_tree.

- [53] Wikipedia. Cross-validation. [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)).
- [54] Wikipedia. Data processing inequality. https://en.wikipedia.org/wiki/Data_processing_inequality.
- [55] Wikipedia. Dirichlet distribution. https://en.wikipedia.org/wiki/Dirichlet_distribution.
- [56] Wikipedia. Errors in variables models. https://en.wikipedia.org/wiki/Errors-in-variables_models.
- [57] Wikipedia. Expectation maximization. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.
- [58] Wikipedia. Gamma function. https://en.wikipedia.org/wiki/Gamma_function.
- [59] Wikipedia. Gated recurrent unit. https://en.wikipedia.org/wiki/Gated_recurrent_unit.
- [60] Wikipedia. Gibbs sampling. https://en.wikipedia.org/wiki/Gibbs_sampling.
- [61] Wikipedia. Hidden Markov model. https://en.wikipedia.org/wiki/Hidden_Markov_model.
- [62] Wikipedia. Importance sampling. https://en.wikipedia.org/wiki/Importance_sampling.
- [63] Wikipedia. Instrumental variables estimation. https://en.wikipedia.org/wiki/Instrumental_variables_estimation.
- [64] Wikipedia. Inverse transform sampling. https://en.wikipedia.org/wiki/Inverse_transform_sampling.
- [65] Wikipedia. Jackknife resampling. [https://en.wikipedia.org/wiki/Cross-validation_\(statistics\)](https://en.wikipedia.org/wiki/Cross-validation_(statistics)).
- [66] Wikipedia. Junction tree algorithm. https://en.wikipedia.org/wiki/Junction_tree_algorithm.
- [67] Wikipedia. k-means clustering. https://en.wikipedia.org/wiki/K-means_clustering.
- [68] Wikipedia. Kalman filter. https://en.wikipedia.org/wiki/Kalman_filter.
- [69] Wikipedia. Kernel method. https://en.wikipedia.org/wiki/Kernel_method.

- [70] Wikipedia. Kernel perceptron. https://en.wikipedia.org/wiki/Kernel_perceptron.
- [71] Wikipedia. Least squares. https://en.wikipedia.org/wiki/Least_squares.
- [72] Wikipedia. Linear regression. https://en.wikipedia.org/wiki/Linear_regression.
- [73] Wikipedia. Long short term memory. https://en.wikipedia.org/wiki/Long_short-term_memory.
- [74] Wikipedia. Markov blanket. https://en.wikipedia.org/wiki/Markov_blanket.
- [75] Wikipedia. Metropolis-Hastings method. https://en.wikipedia.org/wiki/Metropolis%20%93Hastings_algorithm.
- [76] Wikipedia. Minimum spanning tree. https://en.wikipedia.org/wiki/Minimum_spanning_tree.
- [77] Wikipedia. Monte Carlo methods. https://en.wikipedia.org/wiki/Category:Monte_Carlo_methods.
- [78] Wikipedia. Multinomial distribution. https://en.wikipedia.org/wiki/Multinomial_distribution.
- [79] Wikipedia. Multinomial theorem. https://en.wikipedia.org/wiki/Multinomial_theorem.
- [80] Wikipedia. Multivariate normal distribution. https://en.wikipedia.org/wiki/Multivariate_normal_distribution.
- [81] Wikipedia. Natural experiment. https://en.wikipedia.org/wiki/Natural_experiment.
- [82] Wikipedia. Non-negative matrix factorization. https://en.wikipedia.org/wiki/Non-negative_matrix_factorization.
- [83] Wikipedia. Ordinary least squares. https://en.wikipedia.org/wiki/Ordinary_least_squares.
- [84] Wikipedia. Program evaluation and review technique. https://en.wikipedia.org/wiki/Program_evaluation_and_review_technique.
- [85] Wikipedia. Random forest. https://en.wikipedia.org/wiki/Random_forest.
- [86] Wikipedia. Receiver operating characteristic. https://en.wikipedia.org/wiki/Receiver_operating_characteristic.

- [87] Wikipedia. Rejection sampling. https://en.wikipedia.org/wiki/Rejection_sampling.
- [88] Wikipedia. Simple linear regression. https://en.wikipedia.org/wiki/Simple_linear_regression.
- [89] Wikipedia. Simpson's paradox. https://en.wikipedia.org/wiki/Simpson's_paradox.
- [90] Wikipedia. Spring system. https://en.wikipedia.org/wiki/Spring_system.
- [91] Wikipedia. Support vector machine. https://en.wikipedia.org/wiki/Support-vector_machine.
- [92] Wikipedia. Transformer (machine learning model). [https://en.wikipedia.org/wiki/Transformer_\(machine_learning_model\)](https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)).
- [93] Wikipedia. Uplift modelling. https://en.wikipedia.org/wiki/Uplift_modelling.
- [94] Wikipedia. Variational Bayesian methods. https://en.wikipedia.org/wiki/Variational_Bayesian_methods.
- [95] 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>.