ベイズ機械学習入門

担当教授：植野真臣, TA: 木下 涼，菅原聖太

e-mail: kinoshita@ai.is.uec.ac.jp

0.1　スケジュールと成績

- 予定

期日	内容
11 月 21 日（水）13:00 - 14:30	ベイズとコンピュータサイエンス、ビッグデータ、ＡＩ
11 月 21 日（水）14:40 - 16:10	ベイズとコンピュータサイエンス、ビッグデータ、ＡＩ
11 月 28 日（水）13:00 - 14:30	確率、最尤法とベイズ的アプローチ
11 月 28 日（水）14:40 - 16:10	確率、最尤法とベイズ的アプローチ
12 月 3 日（月）13:00 - 14:30	Naïve Bayes classifier の講義
12 月 3 日（月）14:40 - 16:10	Naïve Bayes classifier、NB + Dirichlet の実装
12 月 5 日（水）13:00 - 14:30	TAN、相互情報量などに関する講義
12 月 5 日（水）14:40 - 16:10	TAN の実装
12 月 10 日（月）13:00 - 14:30	TAN + Dirichlet　講義
12 月 10 日（月）14:40 - 16:10	TAN + Dirichlet 実装
12 月 12 日（水）13:00 - 14:30	グラフィカルモデルの講義
12 月 12 日（水）14:40 - 16:10	Bayesian Network の講義
12 月 17 日（月）13:00 - 14:30	Bayesian Network の実装
12 月 17 日（月）14:40 - 16:10	Bayesian Network 学習の実装

- 課題について
  - 注意：絶対に人のソースをコピーしたりしないこと．発覚した場合は不合格になる．
  - 各実装は、Java で行う．
  - データは、すでに形態素解析が行われた文書ファイルを元にして行い、結果とコードを提出する．
  - 途中課題は、研究室のコードをもとに穴埋めの形で実装を行う．
  - 最終課題は分類器などを実装し、自分で見つけたデータに適用して考察等を行う．
- 成績：課題の提出数（＋質）を基準とする．
  ⇒ 最期まですべて実装が正しくできれば優 ⇒ 最後の実験まですべてできれば秀

第1章　確率とビリーフ

1.1　確率

本節では，まず，確率 (probability) を定義する．確率を定義するためには，それが対象とする事象 (event) の集合を定義しなければならない．

定義 1 σ 集合体
Ω を標本空間 *(sample space)* とし，$\mathcal{A}$ が以下の条件を満たすならば σ **集合体** *(σ-field)* と呼ぶ．

1. $\Omega \in \mathcal{A}$
2. $A \in \mathcal{A} \Rightarrow A^c \in \mathcal{A}$ *(ただし，$A^c = \Omega \setminus A$)*
3. $A_1, A_2, \ldots \in \mathcal{A} \Rightarrow \bigcup_{n=1}^{\infty} A_n \in \mathcal{A}$

つまり，互いに素な事象の和集合により新しい事象を生み出すことができ，それら全ての事象を含んだ集合を σ 集合体と呼ぶ．
σ 集合体上で確率 (probability) は以下のように定義される．

定義 2　確率測度
今，σ 集合体 $\mathcal{A}$ 上で，次の条件を満たす測度 *(measure)*P を，**確率測度** *(probability measure)* とよぶ *(Kolmogorov 1933)*.

1. $A \in \mathcal{A}$ について, $0 \leq P(A) \leq 1$
2. $P(\Omega) = 1$
3. 互いに素な事象列 $\{A_n\}_{n=1}^{\infty}$ に対して，

$$P\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} P(A_n)$$

上の定義では,1. は確率が 0 から 1 の値をとること,2. は全事象の確率が 1 になること,3. は互いに素な事象の確率はそれぞれの確率の和で求められること, が示されている．特に 3 の条件を, 確率の加法性 (additivity) と呼ぶ．また，三つ組 $(\Omega, \mathcal{A}, P)$ を確率空間 (probability space) と呼ぶ．

以下，確率の重要な性質を導こう．

定義 1. の 2. より，$P(A^c) = P(\Omega \setminus A) = P(\Omega) - P(A) = 1 - P(A)$ となることがわかる．これより，以下の定理が成り立つ．

定理 1 余事象の確率　事象 A の**余事象** *(complementary event)* の確率は以下のとおりである.

$$P(A^c) = 1 - P(A)$$

また, 定義 1. の 2. より，$P(\phi) = P(\Omega^c) = 1 - P(\Omega) = 1 - 1 = 0$ となる. これより, 以下の定理が成り立つ.

定理 2　境界 $P(\phi) = 0$

さらに，$A \subset B$ のとき,$P(A)$ と $P(B)$ の関係について考えよう.

$A \subset B$ より，$\exists B' : B = A \cup B', A \cap B' = \phi$ が成り立つ. 定義 1. の 3. より，$P(B) = P(A) + P(B')$, 定義 1. の 2. より，　$0 \leq P(B') \leq 1$, が成り立ち，結果として，$P(A) \leq P(B)$ が成り立つことがわかる. これを以下のように**単調性** (monotonicity) と呼ぶ.

定理 3　単調性
$A \subset B$ のとき $P(A) \leq P(B)$

定義 2. の 3. では，互いに素な事象の和の確率はそれぞれの事象の確率の和で求めることができた. では，互いに素ではない事象の和はどのように求められるのであろうか？以下のように求められる.

$A \cup B = (A \cap B^c) \cup (A^c \cap B) \cup (A \cap B)$ より，$P(A \cup B) = P(A \cap B^c) + P(A^c \cap B) + P(A \cap B)$ が成り立つ.

ここで，$P(A \cap B^c) = P(A) - P(A \cap B)$，$P(A^c \cap B) = P(B) - P(A \cap B)$ を代入して，$P(A \cap B^c) + P(A^c \cap B) + P(A \cap B) = P(A) - P(A \cap B) + P(B) - P(A \cap B) + P(A \cap B) = P(A) + P(B) - P(A \cap B)$

これを以下のように**確率の和法則** (Additional low of probability) と呼ぶ.

定理 4　確率の和法則
$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

1.2　主観確率

前節で確率を数学的に定義した. しかし，確率の実際的な解釈には二つの立場がある. 最も一般的な解釈が，ラプラスの**頻度主義**である.

コインを何百回も投げて表が出た回数 (頻度) を数えて，その割合を求めることを考えよう. 今，投げる回数を n, とし，表の出た回数を n_1 とすると,$n \to \infty$ のとき,

$$\frac{n_1}{n} \to \frac{1}{2}$$

となることが予想される. このように, 何回も実験を繰り返して n 回中, 事象 A が n_1 回出たとき，n_1/n を A の確率と解釈するのが頻度主義である.

しかし，この定義では真の確率は無限回実験をしなければ得られないので得ることは不可能である．また，科学的実験が可能な場合にのみ確率が定義され，実際の人間が扱う不確かさに比べて極めて限定的になってしまう．

一方, より広く確率を捉える立場として, 人間の個人的な**主観確率** (subjective probability) として解釈する立場がある．

ベイジアン (Bayesian; ベイズ主義者) は, 確率を主観確率として扱う．次節で導出されるベイズの定理を用いる人々をベイジアンだと誤解されているが, ベイズの定理は確率の基本定理で数学的に議論の余地のないものであり, 頻度主義者も用いる．

例えば, 松原 (2010) では以下のような主観確率の例が挙げられている．

1. 第三次世界大戦が 20XX 年までに起こる確率が 0.01
2. 明日, 会社の株式の価格が上がる確率が 0.35
3. 来年の今日, 東京で雨が降る確率が 0.5

ベイズ統計では, これらの主観確率は個人の意思決定のための信念として定義され, **ビリーフ** (belief) と呼ばれる．当然，頻度論的確率を主観確率の一種とみなすことができるが，その逆は成り立たない．本書では, ベイズ統計の立場に立ち, 確率をビリーフの立場で解釈する．ビリーフの具体的な決定の仕方などは厳密な理論に興味のある読者は Bernardo and Smith(1994), Berger (1985) を参照されたい．

1.3 条件付き確率と独立

本節では，条件付き確率と独立を定義する．

定義 3 条件付き確率
$A \in \mathcal{A}$, $B \in \mathcal{A}$ について，事象 B が起こったという条件の下で，事象 A が起こる確率を**条件付き確率** *(conditional probablity)* と呼び，

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

で示す．

このとき，$P(A|B) = \frac{P(A \cap B)}{P(B)}$ より以下の乗法公式が成り立つ．

定理 5 乗法公式

$$P(A \cap B) = P(A|B)P(B)$$

このとき,$P(A \cap B)$ を A と B の**同時確率** (joint probability) と呼ぶ.

次に，事象の独立を以下のように定義する.

定義 4　独立

ある事象の生起する確率が，他のある事象が生起する確率に依存しないとき，*2* つの事象は**独立** *(independent)* であるという．すなわち事象 A と事象 B が独立とは $P(A|B) = P(A)$ であり，

$$P(A \cap B) = P(A)P(B)$$

が成り立つことをいう.

さらに乗法公式を一般化すると以下のチェーンルールが導かれる.

$$P(A \cap B \cap C) = P(A|B \cap C)P(B \mid C)P(C)$$

これは,3 個以上の事象にも拡張できるので, **チェーンルール** (Chain rule) は以下のように書ける.

定理 6 チェーンルール N 個の事象 $\{A_1, A_2, \ldots, A_N\}$ について

$$P(A_1 \cap A_2 \cap \cdots \cap A_N) = P(A_1|A_2 \cap A_3 \cap \ldots \cap A_N)P(A_2 \mid A_3 \cap A_4 \cap \ldots \cap A_N) \cdots P(A_N)$$

が成り立つ.

1.4　ベイズの定理

本節では，条件付き確率より，ベイズ統計にとって最も重要なベイズの定理を導出する.

ベイズの定理を導出する前に，互いに背反な事象 $A_1, A_2, \cdots, A_n$，$(A_i \in \mathcal{A})$ が全事象 Ω を分割しているとき，事象 $B \in \mathcal{A}$ について以下が成り立つことがわかる.

$$\sum_{i=1}^{n} P(A_i)P(B|A_i) = \sum_{i=1}^{n} P(A_i)\frac{P(A_i \cap B)}{P(A_i)}$$

$$= \sum_{i=1}^{n} P(A_i \cap B) = P(\Omega \cap B) = P(B)$$

これを以下の全確率の定理と呼ぶ.

定理 7　**全確率の定理** *(total probability theorem)*

互いに背反な事象 $A_1, A_2, \cdots, A_n$，$(A_i \in \mathcal{A})$ が全事象 Ω を分割しているとき，事象 $B \in \mathcal{A}$ について, $P(B) = \sum_{i=1}^{n} P(A_i)P(B|A_i)$ が成り立つ.

全確率の定理より，$P(B) = \sum_{i=1}^{n} P(A_i)P(B|A_i)$，従って

$$\frac{P(A_i)P(B|A_i)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)} = \frac{P(A_i)P(B|A_i)}{P(B)} = \frac{P(A_i \cap B)}{P(B)} = P(A_i|B)$$

が成り立つ．これが以下の**ベイズの定理**である．

定理 8　ベイズの定理 *(Bayes' Theorem)*
互いに背反な事象 $A_1, A_2, \cdots, A_n$ が全事象 Ω を分割しているとする．このとき，事象 $B \in \mathcal{A}$ について, $P(A_i|B) = \dfrac{P(A_i)P(B|A_i)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)}$ が成り立つ．

例 1 昔，ある村にうそつき少年がいました．少年はいつも「オオカミが来た !! 」と大声で叫んでいましたが，いままで本当だったことがありません．「オオカミが来た」という事象を A, 少年が「オオカミが来た !! 」と叫ぶ事象を B とし，$P(B \mid A) = 1.0$, $P(B \mid A^c) = 0.5$, $P(A) = 0.005$ としましょう．少年が「オオカミが来た !! 」と叫んだとき実際にオオカミが来ている確率を求めてみよう．

ベイズの定理より，

$$\begin{aligned} P(A|B) &= \frac{P(A)P(B|A)}{P(A)P(B|A) + (1 - P(A))P(B|A^c)} \\ &= \frac{0.005 \times 1.0}{0.005 \times 1.0 + 0.995 \times 0.5} \\ &\approx 0.00995 \end{aligned}$$

すなわち，うそつき少年の情報により，情報がないときに比較して，オオカミの来ている確率が約二倍に上昇していることがわかる．

ベイズ統計では，より広い確率の解釈として「ビリーフ」(belief) を用いることは先に述べた．ここでは考え方のみについて触れよう．意思決定問題から個人的な主観確率であるビリーフが以下のように求められる．

例えば，次の二つの賭けを考えよう．

1. もしオオカミが来ていれば 1 万円もらえる．
2. 赤玉 n 個，白玉 $100 - n$ 個が入っている合計 100 個の玉が入っている壺の中から一つ玉を抜き出し，それが赤玉なら 1 万円もらえる．

どちらの賭けを選ぶかと言われれば，二番目の賭けで赤玉が 100 個ならば，誰もが迷わず二番目の賭けを選ぶだろうし，逆に $n = 0$ ならば，一番目の賭けを選ぶだろう．この二つの賭けがちょうど同等になるように n を設定することができれば，$\frac{n}{100}$ があなたの「オオカミが来る」ビリーフになる．このように，ベイズ統計における確率の解釈「ビリーフ」は頻度主義の確率で扱

える対象を拡張することができ，個人的な信念やそれに基づく意思決定をも合理的に扱えるツールとなるのである．

ビリーフを用いてもう一度例を振り返ろう．例 1 では，もともとのオオカミが来る確率 $P(A) = 0.005$ が，（うそかどうかわからない）少年の報告により $P(A \mid B) = 0.00995$ と約 2 倍にビリーフが更新されていることがわかる．すなわち，うそをつく少年の証言によって事前のビリーフが事後のビリーフに更新されたのである．このとき，ベイズ統計では，少年の証言を「エビデンス」(evidence) と呼び，事前のビリーフを「事前確率」(prior probability)，事後のビリーフを「事後確率」(posterior probability) と呼ぶ．

例 2 *（3 囚人問題）* 次に有名な 3 囚人問題を紹介しよう．ある監獄にアラン，バーナード，チャールズという *3* 人の囚人がいて，それぞれ独房に入れられている．*3* 人は近く処刑される予定になっていたが，恩赦が出て *3* 人のうち *1* 人だけ釈放されることになったという．誰が恩赦になるかは明かされておらず，それぞれの囚人が「私は釈放されるのか*?*」と聞いても看守は答えない．囚人アランは一計を案じ，看守に向かって「私以外の *2* 人のうち少なくとも *1* 人は死刑になるはずだ．その者の名前が知りたい．私のことじゃないんだから教えてくれてもよいだろう*?*」と頼んだ．すると看守は「バーナードは死刑になる」と教えてくれた．それを聞いたアランは「これで釈放される確率が *1/3* から *1/2* に上がった」とひそかに喜んだ．果たしてアランが喜んだのは正しいのか*?*

この問題はベイズの定理を用いれば合理的に解くことができる．アランが釈放される事象を A, バーナードが釈放される事象を B, チャールズが釈放される事象を C とする．今，事前にはだれが釈放されるかわからないので，$P(A) = P(B) = P(C) = \frac{1}{3}$ として良い．また，看守はうそをつかないものとし，看守が証言した事象を E とする．ベイズの定理を用いれば，以下の事後確率を計算すればよい．

$$P(A \mid E) = \frac{P(E \mid A)P(A)}{P(E \mid A)P(A) + P(E \mid B)P(B) + P(E \mid C)P(C)}$$

$P(E \mid A)$ は，アランが釈放されるときに看守はバーナードかチャールズかどちらかの名前を言えたので，看守がバーナードが処刑されるという確率は $\frac{1}{2}$ である．$P(E \mid B)$ は，看守はバーナードが釈放されるときにバーナードが処刑されるとは言わないので 0.0 である．$P(E \mid C)$ は，チャールズが釈放されるときに，アランの名前は言えないので看守はバーナードの名前しか言えず，確率は 1.0 となる．これらの値を上のベイズの定理に挿入すると

$$\begin{aligned} P(A \mid E) &= \frac{P(E \mid A)P(A)}{P(E \mid A)P(A) + P(E \mid B)P(B) + P(E \mid C)P(C)} \\ &= \frac{\frac{1}{2} \cdot \frac{1}{3}}{\frac{1}{2} \cdot \frac{1}{3} + 0.0 \cdot \frac{1}{3} + 1.0 \cdot \frac{1}{3}} = \frac{1}{3} \end{aligned}$$

結果として，アランの釈放される事後確率は事前確率と等しく，看守の証言は情報にはなっていないことがわかる．

以上のようにベイズの定理は帰納推論のための合理的なツールであることがわかる．すなわち，条件付き確率 $P(A \mid B)$ を知っていることは我々が因果 (causality)$B \to A$ を仮定していることになる．従って，ベイズの定理は，因果 (causality)$B \to A$ を用いて A が起こったときの原因の確率 $P(B \mid A)$ を求めるツールである．

1.5 確率変数

一つの試行の結果を**標本点** $\omega \in \Omega$ と呼ぶ．この標本点 $\omega \in \Omega$ は，何らかの測定によって観測される．この測定のことを，確率論では**確率変数** (random variable) と呼ぶ．例えば，コインを n 回投げるという試行について，表が出る回数 X は確率変数である．このとき，標本点 ω は表・裏のパターンが n 個あり得るので 2^n 通りあり，X は 0 から n までの値をとる．

数学的には，確率変数は以下のように定義される．

定義 5 確率空間 $(\Omega, \mathcal{A}, P)$ に対し，Ω から実数 $\mathcal{R}$ への関数 $X : \Omega \to \mathcal{R}$ が，任意の実数 r に対し

$$\{X \leq r\} \in \mathcal{A}$$

を満たすならば，X を確率空間 $(\Omega, \mathcal{A}, P)$ 上の確率変数という．

また，確率変数 X が確率空間 $(\Omega, \mathcal{A}, P)$ 上に定義されると，任意の $r \in \mathcal{R}$ に対して

$$F(r) = P(\{X \leq r\})$$

が定まる．この $F(r)$ は $\mathcal{R}$ から $\mathcal{R}$ への関数であり，**分布関数** (distribution function) と呼ばれる．分布関数 $F(r)$ の性質は，

1. r に対して単調増加
2. $r \to -\infty$ と $r \to \infty$ のとき，それぞれ，極限値 0 と 1 をもつ
3. $F(r)$ は右半連続，すなわち，$h \to 0$ のとき，$F(r+h) - F(r) \to 0$

であり，逆にこれらの性質を満たせば，それを分布関数として持つ確率変数が存在することが知られている．

定義 6 確率変数 X が，高々加算個の実数の集合 $\mathcal{X} = \{x_1, x_2, \ldots, \}$ の中の値をとるならば，X は離散であるという．離散確率変数 X のとりえる値 $x_k \in \mathcal{X}$

を，その確率 $p = P(X = x_k)$ に対応づける写像 $p : \mathcal{X} \to [0, 1]$ を X の**離散確率分布** *(discrete probability distribution)* とよぶ．

離散確率分布 p は

$$\begin{aligned} p(x) &\geq 0 (x \in \mathcal{X}), \\ \sum_{x \in \mathcal{X}} p(x) &= 1 \end{aligned}$$

を満たす．逆に，これら二つの条件を満たす $\mathcal{X}$ 上の関数 p を確率分布としてもつ確率変数が存在する．

定義 7 確率変数 X が，実数全体の集合 $\mathcal{X} = R$ の中の値をとるならば，X は連続であるという．

$$\begin{aligned} f(x) &\geq 0 (x \in R), \\ \int_{-\infty}^{\infty} f(x) &= 1 \end{aligned}$$

を満たし，かつ任意の $a < b$ に対して

$$p(a < X < b) = \int_a^b f(x)dx$$

であるとき，$f(x)$ は連続確率変数 X の**確率密度関数** *(probability density fuction)* であるという．

また，複数の確率変数を持つ確率分布について以下のように定義しよう．

定義 8 今，m 個の確率変数を持つ確率分布 $p(x_1, x_2, \cdots, x_m)$ を変数 $x_1, x_2, \cdots, x_m$ の**同時確率分布** *(joint probability distribution)* と呼ぶ．

同時確率分布から特定の変数の分布を以下のように求めることができる．

定義 9 x_i のみに興味がある場合，同時確率分布から x_i の確率分布は，離散型の場合，

$$p(x_i) = \sum_{x_1, \ldots, x_{i-1}, x_{i+1} \ldots, x_m} p(x_1, x_2, \cdots, x_m),$$

連続型の場合，

$$p(x_i) = \int p(x_1, x_2, \cdots, x_m) dx_1, \ldots, dx_{i-1}, dx_{i+1} \ldots, dx_m,$$

で求められ，$p(x_i)$ を離散型の場合，**周辺確率分布** *(marginal probability distribution)*，連続型の場合，**周辺密度関数** *(marginal probability density function)* と呼ぶ．

1.6 尤度原理

本節では，確率分布のパラメータを定義し，データからパラメータを推定するための尤度原理を紹介する．

定義 10 パラメータ空間と確率分布
k 次元パラメータ集合を $\Theta = \{\theta_1, \theta_2, \cdots, \theta_k\}$ と書くとき，確率分布は以下のような関数で示される．

$$f(x|\Theta)$$

すなわち，確率分布 $f(x|\Theta)$ の形状はパラメータ Θ のみによって決定され，パラメータ Θ のみが確率分布 $f(x|\Theta)$ を決定する情報である．

例 3 コインを n 回投げた時，表が出る回数を確率変数 x とした確率分布は以下の二項分布に従う．

$$f(x|\theta, n) = {}_n\mathrm{C}_x \theta^x (1-\theta)^{n-x}$$

ここで，θ は，コインの表が出る確率のパラメータを示す．

あるデータについて，特定の確率分布を仮定した場合，データからそのパラメータを推定することができる．そのひとつの方法では，以下の尤度を用いる．

定義 11 尤度
$X = (X_1, \cdots, X_i, \cdots, X_n)$ が確率分布 $f(X_i|\theta)$ に従う n 個の確率変数とする．n 個の確率変数に対応したデータ $x = (x_1, \cdots, x_n)$ が得られたとき，

$$L(\theta|x) = \prod_{i=1}^{n} f(x_i|\theta)$$

を**尤度関数** *(Likelihood function)* と定義する *(Fisher,1925)*.

例 4 コインを n 回投げた時，表が出た回数が x 回であったときのコインの表が出るパラメータ θ の尤度は

$$L(\theta|n, x) \propto {}_n\mathrm{C}_x \theta^x (1-\theta)^{n-x},$$

もしくは，

$$L(\theta|n, x) \propto \theta^x (1-\theta)^{n-x}$$

でもよい．

尤度は，データ x パターンが観測される確率に比例する，パラメータ θ の関数である．尤度は確率の定義を満たす保証がないために確率とは呼べないが，これを厳密に確率分布として扱うアプローチが後述するベイズアプローチである．

尤度を最大にするパラメータ θ を求めることは，データ x を生じさせる確率を最大にするパラメータ θ を求めることになり，その方法を**最尤推定法** (Maxmimum Likelihood Estimation; MLE) と呼ぶ．

定義 12 最尤推定量
データ x を所与として，以下の尤度最大となるパラメータを求める時，

$$L(\hat{\theta}|x) = \max\{L(\theta|x) : \Theta \in \mathcal{C}\}$$

$\hat{\theta}$ を**最尤推定量** *(maximum likelihood estimator)* と呼ぶ *(Fisher,1925)*．ただし，$\mathcal{C}$ はコンパクト集合を示す．

例 5　二項分布の最尤推定
コインを投げて n 回中 x 回表が出たときの確率 θ の最尤推定値を求めよう．

コインを n 回投げた時，表が出る回数を確率変数 x とした確率分布は以下の二項分布に従う．

$$f(x|\theta, n) = {}_n\mathrm{C}_x \theta^x (1-\theta)^{n-x}$$

n 回中 x 回表が出たことを所与とした尤度は，

$$L = \prod_{i=1}^{n} f(x_i|\theta)$$

である．${}_n\mathrm{C}_x$ は θ に関係ないので

$$L = \theta^x (1-\theta)^{n-x}$$

ここで，対数尤度 *(Log-likelihood)* を $l = \log L$ とすると，

$$l = x\log\theta + (n-x)\log(1-\theta)$$

$\frac{\partial l}{\partial \theta} = 0$ のとき，l は最大となる．

$$\begin{aligned}\frac{\partial l}{\partial \theta} &= \frac{x}{\theta} - \frac{n-x}{1-\theta} \\ &= \frac{x - x\theta - (n\theta - x\theta)}{\theta(1-\theta)} \\ &= \frac{x - n\theta}{\theta(1-\theta)}\end{aligned}$$

$\frac{\partial l}{\partial \theta} = 0$ とすれば，

$$\therefore \hat{\theta} = \frac{x}{n}$$

すなわち，コインを投げて n 回中 x 回表が出たときの確率の最尤推定値は $\frac{x}{n}$ となる．

例 6 正規分布

$$f(x_i|\mu,\sigma^2)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x_i-\mu)^2}{2\sigma^2}\right\}$$

について，データ $(x_1,x_2,x_3,\cdots,x_n)$ を得たときの平均値パラメータ μ, および分散パラメータ σ^2 の最尤推定値を求めよう．

データ $(x_1,x_2,x_3,\cdots,x_n)$ を得たときの尤度は

$$\begin{aligned}L&=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x_i-\mu)^2}{2\sigma^2}\right\}\\&=\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n\exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\}\end{aligned}$$

ここで，対数尤度を $l=\log L$ とすると

$$l=n\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

$\frac{\partial l}{\partial\mu}=0, \frac{\partial l}{\partial\sigma}=0$ のとき，μ,σ はそれぞれ最大となるので，

$$\frac{\partial l}{\partial\mu}=\sum_{i=1}^{n}\frac{(x_i-\mu)}{\sigma^2}$$

$$\frac{\partial l}{\partial\sigma}=-\frac{n}{\sigma}+\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{\sigma^3}\tag{1.1}$$

$$\begin{aligned}\frac{\partial l}{\partial\mu}&=\frac{\sum x_i-\sum\mu}{\sigma^2}\\&=\frac{\sum x_i-n\mu}{\sigma^2}\end{aligned}$$

$\frac{\partial l}{\partial\mu}=0$ とすれば

$$\hat{\mu}=\frac{\sum_{i=1}^{n}x_i}{n}$$

μ を *(1.1)* 式に代入して，$\frac{\partial l}{\partial\sigma}=0$ とすれば

$$\hat{\sigma^2}=\frac{\sum(x_i-\mu)^2}{n}$$

正規分布での平均値の最尤推定値は $\frac{\sum_{i=1}^{n}x_i}{n}$，分散の最尤推定値は $\frac{\sum(x_i-\mu)^2}{n}$ となる．

推定値の望ましい性質の中で以下の一致性が知られている．

定義 13　強一致性
推定値 $\hat{\theta}$ が真のパラメータ θ^* に概収束する時，$\hat{\theta}$ は**強一致推定値** *(strongly consistent estimator)* であるという．

$$P(\lim_{n\to\infty} \hat{\theta} = \theta^*) = 1.0$$

つまり，データ数が大きくなると推定値が必ず真の値に近づいていくととき，その推定量を強一致推定値と呼ぶ．

このとき，最尤推定値について以下が成り立つ．

定理 9 最尤推定値の一致性
最尤推定値 $\hat{\theta}$ は真のパラメータ θ^* の強一致推定値である．*(Wald, 1949)*

また，一致推定値の漸近的な分布は以下で与えられる．

定義 14
θ^* の推定値 $\hat{\theta}$ が**漸近正規推定量** *(asymptotically normal estimator)* であるとは，$\sqrt{n}(\hat{\theta} - \theta^*)$ の分布が正規分布に分布収束することをいう．すなわち，任意の $\theta^* \in \Theta^*$ と任意の実数に対して

$$\lim_{n\to\infty} P(\frac{\sqrt{n}(\hat{\theta} - \theta^*)}{\sigma(\theta^*)} \leq x) = \Phi(x)$$

このことを，$\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{as} N(0, \sigma^2(\theta^*))$ と書く．$\sigma^2(\theta^*)$ を**漸近分散** *(asymptotic variance)* という．

すなわち，一致推定値の漸近的な分布は正規分布になる．また，一致推定値の誤差は以下のように得られる．

定理 10 確率密度関数が**正則条件**（*Regular condition*）の下で，微分可能のとき，最尤推定量は漸近分散 $I(\theta^*)^{-1}$ を持つ漸近正規推定量である．$I(\theta^*) = E_\theta[(\frac{\partial}{\partial\theta} \ln L(\theta|\boldsymbol{x}))^2]$ を *Fischer* の情報行列と呼ぶ．

1.7　ベイズ推定

前節で述べた尤度原理は古典的な頻度主義であるフィッシャー統計学の流儀である．フィッシャー統計の尤度原理に対して，ベイズ統計では以下の事後分布を用いてパラメータを推定する．

定義 15 事後分布
$X = (X_1, \cdots, X_{\mathrm{n}})$ が独立同一分布 $f(x|\theta)$ に従う n 個の確率変数とする．n 個の確率変数に対応したデータ $x = (x_1, \cdots, x_n)$ が得られた時，

$$p(\theta|x) = \frac{p(\theta)\prod_{i=1}^n f(x_i|\theta)}{\int_\Theta p(\theta)\prod_{i=1}^n f(x_i|\theta)d\theta}$$

を事後分布（*posterior distribution*）と呼び，$p(\theta)$ を事前分布（*prior distribution*）と呼ぶ．

ベイズ統計では，事後分布を最大にするようにパラメータ推定を行う．

定義 16 *MAP* 推定値
データ x を所与として，以下の事後分布最大となるパラメータを求める時，

$$\hat{\theta} = \arg\max\{p(\theta|x) : \theta \in C\}$$

$\hat{\theta}$ を**ベイズ推定値**（*Bayesian estimator*）または，**事後分布最大化推定値:MAP 推定値**（*Maximum A Posterior estimator*）と呼ぶ．
Note:
ベイズ推定は，すべての確率空間で成り立つわけではない．パラメータの事前確率が確率の公理を満たすときにのみ成立する．

また, ベイズ推定値では,MAP 推定値が予測を最適しないことが知られている．予測を最適化するベイズ推定値は, 事後分布を最大化せずに, 事後分布の期待値となる推定値を用いる．

定義 17 *EAP* 推定値
データ x を所与として，以下の事後分布によるパラメータの期待値を求める時，

$$\hat{\theta} = E\{\theta\{p(\theta|x) : \theta \in C\}\}$$

$\hat{\theta}$ を**期待事後推定値:EAP 推定値**（*Expected A Posterior estimator*）と呼ぶ．

ベイズ推定値も強一致性を持つ．

定理 11 ベイズ推定の一致性
ベイズ推定において推定値 $\hat{\theta}$ が真のパラメータ θ^* の強一致推定値となるような事前分布が設定できる

また，ベイズ推定値も漸近的正規性を持ち，誤差を計算できる．

定理 12 ベイズ推定の漸近正規性
事後確率密度関数が正則条件（*Regular condition*）の下で，微分可能のとき，ベイズ推定値が漸近分散 $I(\theta^*)^{-1}$ を持つ漸近正規推定値となる事前分布を設定できる．

ベイズ統計では，どのように事前分布を設定するかが問題となる．事前分布はユーザが知識を十分に持つ場合，自由に決定してよいが，事前に知識を持たない場合にはどのように設定すれば良いのであろうか．このようなときの事前分布を無情報事前分布と呼び，次節のような分布が提案されている．

1.8　無情報事前分布

1.8.1　ジェフリーズの事前分布

事後分布

$$p(\theta|x) = \frac{p(\theta)\prod_{i=1}^{n} f(x_i|\theta)}{\int_\Theta p(\theta)\prod_{i=1}^{n} f(x_i|\theta)d\theta}$$

を求めるための事前分布 $p(\theta)$ の設定について，どのように設定するかが問題となる．通常，データを採取するまで，我々はデータについての情報をもたない．そのために，$p(\theta)$ は無知を表す分布でなくてはならない．このような無知を示す事前分布を**無情報事前分布** (Non-informative prior distribution) と呼ぶ．無知の状態を示す事前分布の選択のルールとして，Jeffreys 1961 は，次の 2 つの提案をしている．まず，1 つの母数 θ について考えると，

1. 母数 θ について，$\theta \in (-\infty, \infty)$ のみの情報があるとき，事前分布は一様分布となる．

$$p(\theta) \propto const$$

2. 母数 θ について，$\theta \in (0, \infty)$ のみの情報があるとき，θ の対数が一様であるような事前分布を考える．すなわち，$p(\log\theta) \propto const$ であるから，変数変換すれば，

$$p(\theta) \propto \frac{1}{\theta}$$

ルール 1 を選択する場合，事後分布＝尤度となるが，$\int_{-\infty}^{\infty} p(\theta) \neq 1$ となり，事前分布 $p(\theta)$ は確率の公理を満たさない．このような事前分布を **improper prior distribution** と呼ぶ．しかし，この improper prior distribution は，ベイズ統計学の整合性を壊すという意味で，議論を招いている．そこで，閉区間に局所的一様分布を考える **principle of stable estimation** (Edwards et al 1963) が提案されている．例えば，$\theta \in [a, b]$ であれば，$p(\theta) = \frac{1}{b-a}$ となり，$\int_{-\infty}^{\infty} p(\theta) = 1$ と確率の公理を満たす．

また，確率変数の定義を満たしたところで，この一様分布の事前分布には問題がある．たとえば，$\theta \in [a, b]$ では，$p(\theta) = const$ であるが，$\kappa = \theta^{10}$ しても，ジェフリーズのルールに従えば，$p(\kappa) = const$ となる．変数変換すれば，そのようにならないことがわかる．このようなことを考慮して，Box and Tiao(1973) は，ある母数 ϕ の尤度が，データが変わってもその形状は変らず，その位置のみを変更させるとき，その母数をデータ移動型母数と呼んだ．以下は，データ移動型母数を見出す方法である．ϕ は対数尤度 $l(\phi|x)$ は，

最尤推定値 $\hat{\theta}$ のまわりでテイラー展開すると，

$$l(\phi|x) = l(\hat{\phi}|x) - \frac{n}{2}(\phi - \hat{\phi})^2 \left(-\frac{1}{n}\frac{\partial^2 l(\phi|x)}{\partial \phi^2} \right)_{\hat{\phi}}$$

ここで

$$I(\theta) = E\left(-\frac{1}{n}\frac{\partial^2 l(\phi|x)}{\partial \phi^2} \right)$$

今，x のデータ発生モデルが指数形分布族であることを仮定する．これは，$\left(-\frac{1}{n}\frac{\partial^2 l(\phi|x)}{\partial \phi^2} \right)_{\phi=\hat{\phi}}$ が $\hat{\phi}$ のみの関数を仮定するのと同値である．

$$J(\hat{\phi}) = \left(-\frac{1}{n}\frac{\partial^2 l(\phi|x)}{\partial \phi^2} \right)_{\phi=\hat{\phi}} = \left(-\frac{1}{n}\frac{\partial^2 l(\theta|x)}{\partial \theta^2} \right)_{\theta=\hat{\theta}} \left(\frac{\partial \theta}{\partial \phi} \right)^2_{\theta=\hat{\theta}} = J(\hat{\theta}) \left(\frac{\partial \theta}{\partial \phi} \right)^2_{\theta=\hat{\theta}}$$

このとき，

$$\left| \frac{\partial \theta}{\partial \phi} \right|_{\theta=\hat{\theta}} \propto J^{-1/2}(\hat{\theta})$$

となるように変換 ϕ を選べば， $J(\hat{\phi})$ は定数となり，尤度は $(\phi - \hat{\phi})^2$ の関数となる．すなわち，ϕ に関して近似的データ移動型となる．このとき，無情報事前分布は

$$p(\theta) \propto \left| \frac{\partial \theta}{\partial \phi} \right|_{\theta=\hat{\theta}} \propto J^{-1/2}(\hat{\theta})$$

となる．

また，指数型分布の仮定を抜いた場合，

$$\left| \frac{\partial \theta}{\partial \phi} \right|_{\theta=\hat{\theta}} \propto I^{-1/2}(\theta)$$

となる． $I(\theta)$ はフィッシャー情報量を示す．

すなわち，母数 θ の事前分布は，フィッシャー情報量 $I(\theta)$ に比例させるというルールである．これが，有名なジェフリーズの事前分布である (Jeffrys 1961)．

データ情報最大化事前分布（maximum data information distribution）

Zellner(1971) は，データのもつ情報と比較して，事前情報のもつ情報を最小にするような分布を無情報事前分布としている．

情報を情報理論の枠組みで定義すると，事前分布における情報量と事後分布における情報量との差として伝達情報量で定義することができる．

すなわち，

$$G = \int_{\Theta} p(\theta) \log \theta d\theta - \int_x \int_{\Theta} p(x|\theta) \log p(x|\theta) d\theta p(x) dx$$

を最大化させる事前分布を，**データ情報最大化事前分布**（maximum data information distribution）と呼ぶ．

1.8.2　自然共役事前分布（natural conjugate prior distribution）

ベイズ統計の中で最も一般的で，ベイズ的な有効性を発揮できると考えられるのが，この自然共役事前分布である．先までの事前分布では，データを得る前の事前分布とデータを得た後の事後分布は，分布の形状が変化する．しかし，データの有無に係らず，分布の形状は同一のほうが自然であろう．そこで，事前分布と事後分布が同一の分布族に属する時，その事前分布を**自然共役事前分布**（natural conjugate prior distribution）と呼ぶ．ここでは，特にこの自然共役事前分布を中心にベイズ的推論を行なうようにする．

例 7 二項分布

$$f(x|\theta,n) = {}_nC_x\theta^x(1-\theta)^{n-x}$$

コインを投げて n 回中 x 回表が出たときの確率 θ をベイズ推定しよう．

尤度関数は，${}_nC_x\theta^x(1-\theta)^{n-x}$ であり，二項分布の自然共役事前分布は，以下の**ベータ分布** $(Beta(\alpha,\beta))$ である．

$$p(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

事後分布は，

$$p(\theta|n,x,\alpha,\beta) = \frac{\Gamma(n+\alpha+\beta)}{\Gamma(x+\alpha)\Gamma(n-x+\beta)}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}$$

とやはりベータ分布となる．

対数をとり，以下の対数事後分布を最大化すればよい．

$$\begin{aligned} & \log p(\theta|n,x,\alpha,\beta) \\ = \quad & \log\frac{\Gamma(n+\alpha+\beta)}{\Gamma(x+\alpha)\Gamma(n-x+\beta)} + (x+\alpha-1)\log\theta + (n-x+\beta-1)\log(1-\theta) \end{aligned}$$

$\frac{\partial \log p(\theta|n,x,\alpha,\beta)}{\partial\theta} = 0$ のとき，対数事後分布は最大となるので，

$$\begin{aligned} & \frac{\partial \log p(\theta|n,x,\alpha,\beta)}{\partial\theta} \\ = \quad & \frac{x+\alpha-1}{\theta} - \frac{n-x+\beta-1}{1-\theta} = \frac{x+\alpha-1-x\theta-\alpha\theta+\theta-n\theta+x\theta-\beta\theta+\theta}{\theta(1-\theta)} \\ = \quad & \frac{x+\alpha-1-(n+\alpha+\beta-2)\theta}{\theta(1-\theta)} = 0 \end{aligned}$$

$\theta(1-\theta) \neq 0$ より

$$\theta = \frac{x+\alpha-1}{n+\alpha+\beta-2}$$

がベイズ推定値となる．さて，α,β は事前分布のパラメータであるが，これを**ハイパーパラメータ**（*Hyper parameter*）と呼ぶ．このハイパーパラメータによって，事前分布は様々な形状をとる．

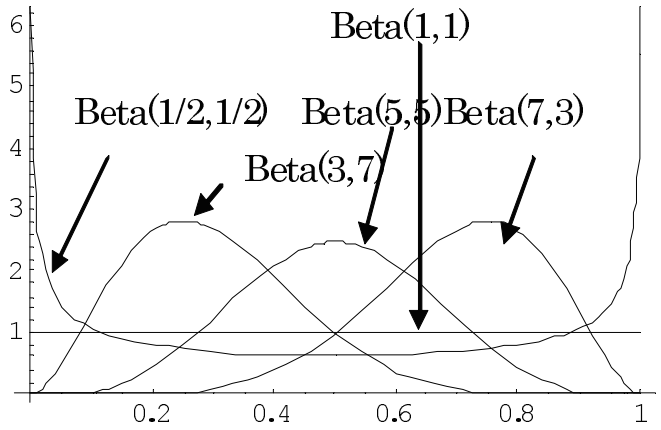


図 1.1: ハイパーパラメータと事前分布の形状

例えば，事前分布が一様となる場合の推定値は，$\widehat{\theta} = \frac{x}{n}$ となり，最尤解に一致する．

例 8　正規分布

$$P(x_i|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x_i-\mu)^2}{2\sigma^2}\right\}$$

$(x_1, x_2, \cdots, x_n)$ を得たときの μ, σ^2 を求めよう．

尤度は，

$$\begin{aligned} L &= \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x_i-\mu)^2}{2\sigma^2}\right\} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\} \end{aligned}$$

このとき，自然共役事前分布は，

$$p(\mu) = N(\mu_0, \sigma_0^2)$$

$$p(\sigma^2) = \chi^{-2}(\nu_0, \lambda_0)：\text{逆カイ二乗分布}$$

すなわち，事前分布はこれらの積の形で以下のように表される．

$$\begin{aligned} p(\mu,\sigma^2) = p(\mu|\sigma^2)p(\sigma^2) &\propto \left(\frac{\sigma^2}{n_0}\right)^{-\frac{1}{2}}\exp\left\{-\frac{n_0(\mu-\mu_0)^2}{2\sigma^2}\right\}(\sigma^2)^{-\frac{1}{2}\nu_0-1}\exp\left(-\frac{\lambda_0}{2\sigma^2}\right) \\ &= (\sigma^2)^{-\frac{1}{2}(\nu_0+1)-1}\exp\left\{-\frac{\lambda_0+n_0(\mu-\mu_0)^2}{2\sigma^2}\right\} \end{aligned}$$

ここで， $n_0, \mu_0, \nu_0, \lambda_0$ はハイパーパラメータであり，$n_0 = \nu_0 + 1$ という関係にある．

一方，これを尤度に掛け合わせて事後分布を導くのであるが，計算の簡便さのために，以下のように尤度を変形させる．

$$L = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\}$$

ここで指数部分 $\exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\}$ を三平方の定理により，推定平均 $\bar{x}$ を介して，以下のように分解する．

$$\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2} = \sum_{i=1}^{n}\left[\frac{(x_i-\bar{x})^2}{2\sigma^2} + \frac{(\bar{x}-\mu)^2}{2\sigma^2}\right]$$

これより，尤度 L は，

$$\begin{aligned} L &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\bar{x})^2}{2\sigma^2}\right\}\exp\left\{-\frac{n(\bar{x}-\mu)^2}{2\sigma^2}\right\} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\frac{S^2+n(\mu-\bar{x})^2}{2\sigma^2}\right\} \end{aligned}$$

ただし，ここで，$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $S^2 = \sum_{i=1}^{n}(x_i-\bar{x})^2$ と書き換えられる．

さて，この尤度 L と先の事前分布を掛け合わせることによって，以下のような事後分布が得られる．ここで，$\nu_0 = n_0 - 1$ とおいて，

$$\begin{aligned} p(\mu,\sigma^2|X) \propto L \times p(\mu,\sigma^2) &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left\{-\frac{S^2+n(\mu-\bar{x})^2}{2\sigma^2}\right\} \\ &\quad \times(\sigma^2)^{-\frac{1}{2}(\nu_0+1)-1}\exp\left\{-\frac{\lambda_0+n_0(\mu-\mu_0)^2}{2\sigma^2}\right\} \\ &\propto (\sigma^2)^{-\frac{1}{2}(n+n_0)-1}\exp\left\{-\frac{\lambda_0+S^2+n_0(\mu-\mu_0)^2+n(\mu-\bar{x})^2}{2\sigma^2}\right\} \end{aligned}$$

さらに，指数部分のうち，$\lambda_0 + S^2$ 以外の部分に，平方完成を行なうと，結局，

$$p(\mu,\sigma^2|X) \propto (\sigma^2)^{-\frac{1}{2}(n+n_0)-1}\exp\left\{-\frac{\lambda_* + (n_0+n)(\mu-\mu_*)^2}{2\sigma^2}\right\}$$

ただし，

$$\lambda_* = \lambda_0 + S^2 + \frac{n_0 n(\bar{x}-\mu_0)^2}{n_0+n}$$

$$\mu_* = \frac{n_0\mu_0 + n\bar{x}}{n_0+n}$$

となる．この事後分布もまた，正規分布と逆カイ二乗分布の積となり，$N \times \chi^{-2}(n_0+n, \mu_*, \nu_0+n, \lambda_*)$ と略記する．

さて，これらの事後分布は，μ と σ^2 の同時事後確率分布であることがわかる．このように，複数のパラメータを同時に最大化させる場合，次のような周辺化（*marginalization*）を行い，個々のパラメータの分布を導く．このような分布を周辺事後分布（*marginal posterior distribution*）と呼ぶ．

すなわち，パラメータ μ についての周辺事後分布は以下のようにして求められる．

$$\begin{aligned} p(\mu|X) &= \int_0^\infty p(\mu,\sigma^2|X)p(\sigma^2)d\sigma^2 \\ &\propto \frac{\Gamma[(\nu_*+1)/2]}{\sqrt{\nu_*\pi\lambda_*/n_*}\Gamma(\nu_*/2)}\left[1+\frac{(\mu-\mu_*)^2}{\mu_*}\right]^{-(\nu_*+1)/2} \\ &\equiv t(\nu_*,\mu_*,\lambda_*/n_*) \end{aligned}$$

このように μ の周辺事後分布は，t 分布 $t(\nu_*,\mu_*,\lambda_*/n_*)$ に従うことがわかる．

また，パラメータ σ^2 についての周辺事後分布も同様にして，以下のように求められる．

$$\begin{aligned} p(\sigma^2|X) &= \int_{-\infty}^\infty p(\mu,\sigma^2|X)p(\mu)d\mu \\ &\propto \frac{\lambda_*^{\nu_*/2}}{2^{\nu_*/2}\Gamma(\nu_*/2)}(\sigma^2)^{-\nu_*/2-1}\exp\left[-\frac{\lambda_*}{2\sigma^2}\right] \\ &\equiv \chi^{-2}(\nu_*,\lambda_*) \end{aligned}$$

となり，σ^2 の周辺事後分布は，逆カイ二乗分布 $\chi^{-2}(\nu_*,\lambda_*)$ に従うことがわかる．

また，事後確率最大化によるベイズ推定値は，t 分布のモードが，μ_* であることより，μ の推定値は，

$$\hat{\mu} = \frac{n_0\mu_0 + n\bar{x}}{n_0+n}$$

となり，σ^2 のベイズ推定値は，逆カイ二乗分布のモードが，$\frac{\lambda_*}{(\nu_*-2)}$ であることより，σ^2 の推定値は，

$$\hat{\sigma^2} = \frac{\left[\lambda_0 + S^2 + \frac{n_0 n(\bar{x}-\mu_0)^2}{n_0+n}\right]}{\nu_*-2}$$

となる．

1.9 予測分布 (predictive distribution)

データやモデルを用いて推論を行なう重要な目的の一つに，未知の事象の予測が挙げられる．この予測問題のためには，最もよく用いられるのは，$p(y|\hat{\theta})$ で示される plug-in distribution と呼ばれる分布である．しかし，$\hat{\theta}$ は推定値

であるためにそのサンプルのとり方によってこの分布は大きく変化する．ベイズ的アプローチでは，この $\hat{\theta}$ のばらつき（$\hat{\theta}$ の事後分布）を考慮し，以下のように予測分布を定義する．

定義 18
モデル m から発生されるデータ x により，未知の変数 y の分布を予測する時，以下の分布を**予測分布** *(Predictive distribution)* と呼ぶ．

$$p(y|x,m) = \int_{\Theta} p(y|\theta,m)p(\theta|x,m)d\theta$$

例 9 二項分布
ベータ分布を事前分布とした二項分布の予測分布は，以下のようになる．

$$\begin{aligned}
p(y|x) &= \int_{\Theta} p(y|\theta)p(\theta \mid x)d\theta \\
&= \int_{\Theta} {}_nC_y\theta^y(1-\theta)^{n-y}\frac{\Gamma(n+\alpha+\beta)}{\Gamma(x+\alpha)\Gamma(n-x+\beta)}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1}d\theta \\
&\propto {}_nC_y\frac{\Gamma(y+1)\Gamma(n-y+1)}{\Gamma(n+2)}\frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(n+\alpha+\beta)} \\
&= \frac{n!}{y!(n-y)!}\frac{\Gamma(y+1)\Gamma(n-y+1)}{\Gamma(n+2)}\frac{\Gamma(x+\alpha)\Gamma(n-x+\beta)}{\Gamma(n+\alpha+\beta)}
\end{aligned}$$

特に，α, β が整数のとき

$$p(y|x) \propto \frac{n!}{y!(n-y)!}\frac{y!(n-y)!}{(n+1!)}\frac{(x+\alpha-1)!(n-x+\beta-1)!}{(n+\alpha+\beta-1)!}$$

例 10 正規分布
事前分布を $N(\mu,\sigma^2)$ 分布

$$\begin{aligned}
p(\mu,\sigma^2) &= p(\mu|\sigma^2)p(\sigma^2) \propto \left(\frac{\sigma^2}{n_0}\right)^{-\frac{1}{2}}\exp\left\{-\frac{n_o(\mu-\mu_0)^2}{2\sigma^2}\right\}(\sigma^2)^{-\frac{1}{2}\nu_0-1}\exp\left(-\frac{\lambda_0}{2\sigma^2}\right) \\
&= (\sigma^2)^{-\frac{1}{2}(\nu_0+1)-1}\exp\left\{-\frac{\lambda_0+n_0(\mu-\mu_0)^2}{2\sigma^2}\right\}
\end{aligned}$$

とし，尤度は

$$\begin{aligned}
L &= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n\exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}\right\} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n\exp\left\{-\sum_{i=1}^{n}\frac{(x_i-\bar{x})^2}{2\sigma^2}\right\}\exp\left\{-\frac{n(\bar{x}-\mu)^2}{2\sigma^2}\right\} \\
&= \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n\exp\left\{-\frac{S^2+n(\mu-\bar{x})^2}{2\sigma^2}\right\}
\end{aligned}$$

であるから，事後分布は

$$\begin{aligned}
&p(\mu,\sigma^2|X) \\
&\propto (\sigma^2)^{-\frac{1}{2}(n+n_0)-1}\exp\left\{-\frac{\lambda_*+(n_0+n)(\mu-\mu_*)^2}{2\sigma^2}\right\}
\end{aligned}$$

ただし，

$$\lambda_* = \lambda_0 + S^2 + \frac{n_0 n(\bar{x} - \mu_0)^2}{n_0 + n}$$

$$\mu_* = \frac{n_0\mu_0 + n\bar{x}}{n_0 + n}$$

となる．

予測分布は

$$p(x_{n+1}|X) = \int\int p(x_{n+1}|\mu,\sigma^2)p(\mu,\sigma^2|x_1,\cdots,x_n)d\mu d\sigma^2$$

ここで，

$$p(x_{n+1}|\mu,\sigma^2) \propto (\sigma^2)^{-1}\exp\left[-\frac{(x_{n+1}-\mu)^2}{2\sigma^2}\right]$$

より，

$$\begin{aligned}
p(x_{n+1}|X) &= \int\int p(x_{n+1}|\mu,\sigma^2)p(\mu,\sigma^2|x_1,\cdots,x_n)d\mu d\sigma^2 \\
&\propto \int\int (\sigma^2)^{-\frac{\nu+1}{2}-2}\exp\left[-\frac{(x_{n+1}-\mu)^2 + S^2 + n(\mu-\bar{x})^2}{2\sigma^2}\right]d\mu d\sigma^2 \\
&= \int\int (\sigma^2)^{-\frac{\nu+1}{2}-2}\exp\left[-\frac{1}{2\sigma^2}\left((n+1)(\mu-\bar{\mu})^2 + S^2 + \frac{n}{n+1}(x_{n+1}-\bar{x})^2\right)\right]d\mu d\sigma^2 \\
&\propto \int (\sigma^2)^{-\frac{\nu+1}{2}-1}\exp\left[-\frac{1}{2\sigma^2}\left(S^2 + \frac{n}{n+1}(x_{n+1}-\bar{x})^2\right)\right]d\sigma^2 \\
&\propto \left(S^2 + \frac{n}{n+1}(x_{n+1}-\bar{x})^2\right)^{-\frac{\nu+1}{2}} \\
&\propto \left(1 + \left[\frac{(x_{n+1}-\bar{x})}{\sqrt{\frac{n+1}{n\nu}S^2}}\right]^2/\nu\right)^{-\frac{\nu+1}{2}}
\end{aligned}$$

ただし，ここで

$$\bar{\mu} = \frac{n\bar{x} + x_{n+1}}{n+1}$$

ここで， $t = \frac{(x_{n+1}-\bar{x})}{\sqrt{\frac{n+1}{n\nu}S^2}}$ とおくとき，t は自由度 ν の t 分布に従う．

1.10 周辺尤度 (marginal likelihood)

モデルの候補が複数ある場合に，データ x からモデル m を選択することを**モデル選択**（model selection）と呼ぶ．ベイズ統計では，一般的に，モデル選択のために以下の周辺尤度を最大にするモデルを選択する．

定義 19
データ x を所与としたモデル m の尤度を周辺化して**周辺尤度** *(marginal likelihood)* と呼ぶ.

$$p(x|m) = \int_{\Theta} p(x|\theta, m)p(\theta|m)d\theta \tag{1.2}$$

一般的に，周辺尤度は事後分布についてパラメータ空間を積分しなければならないので計算が難しい．モデルを一般化して近似的に対数周辺尤度を求めたものが BIC(Bayesian Information Criterion) である.

定義 20
データ x を所与としたモデル m の対数周辺尤度は以下の *BIC(Bayesian Information Criterion)* で求められる.

$$\log p(x|m) \approx BIC = \log p(x|\theta, m) - \frac{K}{2}\log n, \tag{1.3}$$

ここで K はモデルのパラメータ数，n はデータ数を示す.

第2章　ベイズ分類機

1 章まででベイズの基本を学んできた．本章から本実験のテーマであるベイズ分類機を学ぶ．

ベイズ分類機は次のように定義される．

定義 21 確率変数集合 $\mathbf{X} = \{X_1, \cdots, X_n\}$ の各変数がそれぞれ $x_1, \cdots, x_n$ の値を持つ（インスタンス化された）時，離散確率変数 $X_0 = \{x_0\} = \{0, 1, \cdots, r_0\}$ の $\mathbf{X}$ による分類問題とは，以下の $\widehat{x}_0$ を求めることである．

$$\widehat{x}_0 = \underset{x_0}{\operatorname{argmax}}\, p(x_0 \mid x_1, \cdots, x_n) \tag{2.1}$$

ここで，X_0 を目的変数，$X_i \in \mathbf{X}, (i = 1, \cdots, n)$ をその説明変数と呼ぶ．機械学習では，$X_i \in \mathbf{X}, (i = 1, \cdots, n)$ を特徴量と呼ぶことが多い．

式 (2.1) の $p(x_0 \mid x_1, \cdots, x_n)$ はベイズの定理により，以下のように求められる．

$$\begin{aligned} \underset{x_0}{\operatorname{argmax}}\, p(x_0 \mid x_1, \cdots, x_n) &= \underset{x_0}{\operatorname{argmax}}\, \frac{p(x_0)p(x_1, \cdots, x_n \mid x_0)}{p(x_1, \cdots, x_n)} \\ &= \underset{x_0}{\operatorname{argmax}}\, p(x_0)p(x_1, \cdots, x_n \mid x_0) \end{aligned} \tag{2.2}$$

このとき，$p(x_1, \cdots, x_n \mid x_0)$ はモデルのデータ $\{x_1, \cdots, x_n\}$ に対する尤度に対応し，式 (2.1) を識別関数と呼ぶ．

ベイズ分類機の尤度 $p(x_1, \cdots, x_n \mid x_0)$ の計算法は仮定するモデルによってさまざまに変化する．モデルの制約が強い場合（単純なモデルの場合）は計算が容易であるが，モデルが複雑になるに従い計算も複雑になる．以下、このモデルを単純なものから徐々に一般化して学んでいくことにしよう．

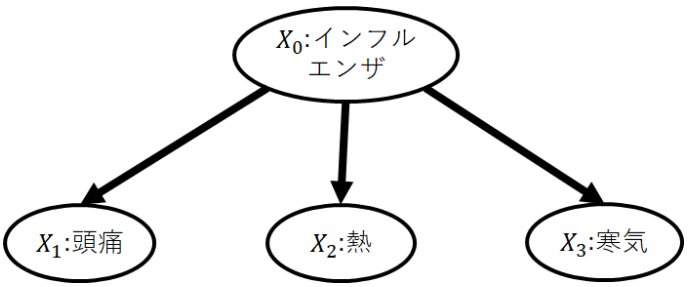


図 2.1: インフルエンザ判別に用いられる Naive Bayes の例

2.1 Naive Bayes

まず最初に最も単純な構造をもつベイズ分類器である Naive Bayes を学ぼう.

Naive Bayes では，図 2.1 のように，目的変数が与えられた際，説明変数間の条件付き独立を仮定している．これにより, 同時確率分布を以下のように，単純な確率の積で表すことができる.

$$p(X_0, X_1, ..., X_n) = p(X_0) \prod_{i=1}^{n} p(X_i \mid X_0)$$

ここで，$p(X_0 = x_0)$ $(x_0 = 1, \cdots, r_0)$，$p(X_i = k \mid X_0 = x_0)$ $(i = 1, \cdots, n; x_0 = 1, \cdots, r_0; k = 1, \cdots, r_i)$ を示すパラメータをそれぞれ $\theta_{x_0}, \theta_{ix_0k}$ で表すと，説明変数のデータ（インスタンス）$x_1, \cdots, x_n$ に対する Naive Bayes の識別関数は以下で表される.

$$\widehat{x}_0 = \underset{x_0}{\operatorname{argmax}}\, \theta_{x_0} \prod_{i=1}^{n} \theta_{ix_0x_i} \tag{2.3}$$

識別関数は、非常に少ないパラメータを元に計算することが可能であり，モデル全体のパラメータ数は変数数に対して線形関数的に増加するに留まる.

今，全変数がインスタンス化されたサンプルが N 個あり，t 番目のサンプルを $\mathbf{d}^t = \langle x_0^t, x_1^t, \cdots, x_n^t \rangle$ と表し，訓練データを $D = \langle \mathbf{d}^1, \cdots, \mathbf{d}^t, \cdots, \mathbf{d}^N \rangle$ と表すと，D を所与とした時の Naive Bayes の尤度は以下で表される.

$$\prod_{x_0=1}^{r_0} \theta_{x_0}^{N_{x_0}} \prod_{i=1}^{n} \prod_{x_0=1}^{r_0} \prod_{k=1}^{r_i} \theta_{ix_0k}^{N_{ix_0k}} \tag{2.4}$$

ここで，N_{x_0} は D において $X_0 = x_0$ となる頻度を表し，N_{ix_0k} は D において $X_0 = x_0$ のときの $X_i = k$ となる頻度を表す．さらに，Naive Bayes の最尤推定量は以下のように表される.

$$\hat{\theta}_{x_0} = \frac{N_{x_0}}{N}, \quad \hat{\theta}_{ix_0k} = \frac{N_{ix_0k}}{N_{x_0}} \tag{2.5}$$

2.1.1 ゼロ頻度問題

テストデータの中で，訓練データに含まれないデータを 1 つでも含んでいると，識別関数の計算に用いるパラメータ θ_{x_0} や θ_{ix_0k} の最尤推定値が 0 となってしまうことがある．この場合，識別関数の値も 0 となり（対数のときは $\log 0$ となり計算できない），そのカテゴリの確率は 0 になってしまう.

たとえばスパムメール分類を行うとき，メール文に ”無料”, ”儲かる ”などの単語が含まれており,「これはスパムメールである確率が高くなってきた」と思っていても，訓練時には含まれなかった新単語”稼ぐ”が出現したとする.

そうすると，そのパラメータは 0 と推定されるため，このメールがスパムである確率は 0 になってしまう．この問題は，ゼロ頻度問題と呼ばれており，スムージングという方法で緩和できる．旧来の統計学でよく用いられるのが単語の出現回数に 1 を加えるラプラススムージング（Laplace Smoothing）である．新しい単語が出てくると確率は低くなるが，0 にはならないため，問題なく推定することが可能になる．

演習問題 1

Naive Bayes のパラメータの最尤推定量 $\hat{\theta}_{x_0}, \hat{\theta}_{ix_0k}$ が上の値になることを確かめよ．
ヒント：束縛条件 $\sum_{x_0=1}^{r_0} \theta_{x_0} = 1$，$\sum_{k=1}^{r_i} \theta_{ix_0k} = 1$ を含んだラグランジュの未定乗数法を用いる．

演習問題 2

以下の”NB.java”内の関数”getParameters”，”classification”，”setFrequencyTable”を実装し，Naive Bayes による分類プログラムを完成させよ．NB.java がわかりにくい人は，Naive Bayes の分類プログラムを一から作成しても良い．
また，データセット”spam”に対して Naive Bayes による分類精度を求めよ．
※ NB.java 及びデータセット”spam”は http://www.ai.lab.uec.ac.jp/実験/にある MICS_NB_TAN プロジェクト内にある．
※ eclipse プロジェクトのインポート方法は
http://www.ai.lab.uec.ac.jp/wp-content/uploads/2018/11/cabebc8b347bc1e77dcdbf08de59ff4c.pdf を参照してください．
※本実験で公開しているデータセットでは，最も右側の列を目的変数の列としている．

ソースコード 2.1: NB.java

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileReader;
4  import java.io.IOException;
5  import java.util.ArrayList;
6  import java.util.HashMap;
7  public class NB {
8    // Number of Categories
9    int NC = 2;
10   // Number of Variables
11   int NV;
12   // Leaning Data
13   ArrayList<int[]> LD;
```

```
14  // Test Data
15  ArrayList<int[]> TD;
16  //頻度表
17  double[] ft_c;
18  ArrayList<double[][]> ft_ic;
19  public NB() throws IOException{
20    //データセット入力
21    String dataset = "banknote";
22    System.out.println("Dataset: "+dataset);
23    //学習データ読み込み
24    readLearningData("data/"+dataset+"/LD.csv");
25    //頻度表の生成
26    setFrequencyTable();
27    //テストデータ読み込み
28    readTestingData("data/"+dataset+"/TD.csv");
29    //NB のパラメータ学習
30    ArrayList<HashMap<Integer, Double>> parameters =
          getParameters();
31    //学習したNB でテストデータを分類
32    classification(parameters);
33  }
34
35  // 変数自身とその親の値を鍵としてその条件付き確率パラメータ
      を返すような
      HashMap を，各変数ごとに ArrayList で繋いだものを返す関数
36  private ArrayList<HashMap<Integer, Double>> getParameters
      (){
37
38
39
40
41
42
43  }
44
45  //パラメータを用いてテストデータの分類をし,その正答率を表示
      する.
46  private void classification(ArrayList<HashMap<Integer,
      Double>> parameters){
47    double num_correct_prediction = 0;
48    for(int d = 0; d < TD.size(); ++d){
49
50
51
52
53
```

```
54
55        //正答回数測定
56        if(predictedClass == TD.get(d)[NV - 1]){
57          num_correct_prediction++;
58        }
59      }
60      //正答率表示
61      System.out.println("Classification accuracy: "+
            num_correct_prediction / (double)TD.size());
62    }
63
64    // 訓練データ読み込み
65    private void readLearningData(String DataBaseName) throws
          IOException{
66      BufferedReader br = new BufferedReader(new FileReader(new
            File(DataBaseName)));
67      ArrayList<String[]> tLD = new ArrayList<String[]>();
68      String line = br.readLine();
69      NV = line.split(",").length;
70      while(line != null){
71        String[] data = line.split(",").clone();
72        tLD.add(data);
73        line = br.readLine();
74      }
75      LD = new ArrayList<int[]>();
76      for(int i = 0; i < tLD.size(); ++i){
77        LD.add(new int[NV]);
78        for(int j = 0; j < NV; ++j){
79          LD.get(i)[j] = Integer.parseInt(tLD.get(i)[j]);
80        }
81      }
82      br.close();
83    }
84
85    // テストデータ読み込み
86    private void readTestingData(String DataBaseName) throws
          IOException{
87      BufferedReader br = new BufferedReader(new FileReader(new
            File(DataBaseName)));
88      ArrayList<String[]> tTD = new ArrayList<String[]>();
89      String line = br.readLine();
90      while(line != null){
91        String[] data = line.split(",").clone();
92        tTD.add(data);
93        line = br.readLine();
94      }
```

```
95      TD = new ArrayList<int[]>();
96      for(int i = 0; i < tTD.size(); ++i){
97        TD.add(new int[NV]);
98        for(int j = 0; j < NV; ++j){
99          TD.get(i)[j] = Integer.parseInt(tTD.get(i)[j]);
100       }
101     }
102     br.close();
103   }
104
105   //N_{x_0}とN_{i x_0 k}をそれぞれft_c, ft_ic に格納する関数
106   private void setFrequencyTable(){
107     //初期化
108     ft_c = new double[NC];
109     ft_ic = new ArrayList<>();
110     for(int i = 0; i < NV - 1; ++i){
111       ft_ic.add(new double[NC][NC]);
112     }
113     //ft_c, ft_ic を計算
114     for(int d = 0; d < LD.size(); ++d){
115
116
117
118
119     }
120   }
121
122   public static void main(String[] args) throws IOException {
123     new NB();
124   }
125 }
```

2.2 Tree Augmented Naive Bayes

前節で紹介した Naive Bayes は，各説明変数が目的変数を所与として条件付き独立であることを仮定している．しかし，一般にこの仮定は成り立たない．例えば図.1 において，一般に熱があった場合は，インフルエンザの感染に関係なく悪寒がする確率が高まるため，明らかに二つの変数 X_1 と X_2 は X_0 を所与として従属関係である．しかし，Naive Bayes では X_1 と X_2 を X_0 を所与として条件付き独立としているため，誤った確率を推定してしまう．この問題を解決するには，従属関係にある説明変数間に適切にエッジを引く必要がある．しかし，全ての説明変数同士の従属関係をチェックするには，膨大な計算時間がかかってしまう．そこで，説明変数間の従属関係を考慮し，かつ

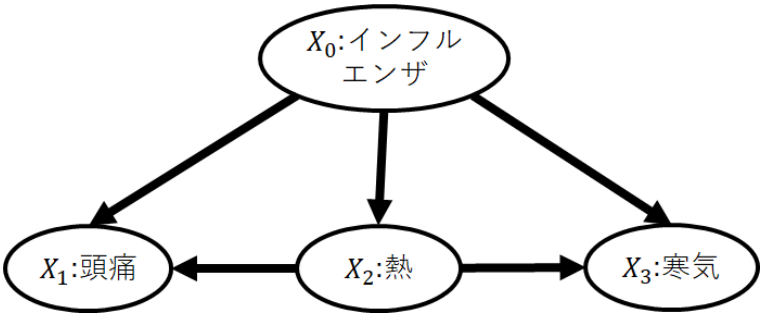


図 2.2: インフルエンザ判別に用いられる TAN の例

学習の計算量が少ないモデルとして，Tree Augmented Naive Bayes (TAN) が提案されている．TAN は，Naive Bayes のように目的変数が各説明変数の親となっており，全説明変数とそれらを結ぶエッジが木構造となっているモデルである（図 2.2）．

TAN は同時確率分布を次のように表す．

$$p(X_0, X_1, \cdots, X_n) = p(X_0) \prod_{i=1}^{n} p(X_i \mid \Pi_i)$$

ここで，Π_i は TAN における変数 X_i の親変数集合である．TAN では各説明変数が目的変数を親として持つので，$X_0 \in \Pi_i, (i = 1, \cdots, n)$ である．

今，Π_i が j 番目のパターンをとる $(\Pi_i = j)$ ときに $X_i = k$ となる条件付き確率 $p(X_i = k \mid \Pi_i = j), (i = 1, \cdots, n; j = 1, \cdots, q_i; k = 1, \cdots, r_i)$ を示すパラメータを θ_{ijk} と表すと，説明変数のインスタンス $x_1, \cdots, x_n$ に対する TAN の識別関数は次式で表される．

$$\widehat{x}_0 = \underset{x_0}{\operatorname{argmax}}\, \theta_{x_0} \prod_{i=1}^{n} \theta_{i\pi_i x_i} \tag{2.6}$$

ここで，π_i はインスタンスにおいて X_i の親変数集合 Π_i がとっている状態を表す．また，D を所与とした TAN の尤度は以下で表される．

$$\prod_{x_0=1}^{r_0} \theta_{x_0}^{N_{x_0}} \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}} \tag{2.7}$$

この尤度に対し，TAN のパラメータ θ_{ijk} の最尤推定量は以下で表される．

$$\widehat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

ここで，$N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ である．

TAN の説明変数が成す木構造は未知であるから，データから学習する必要がある．今，TAN の全パラメータ集合を Θ とし，TAN のとりうる全グラフ

集合を $\mathcal{G}_{TAN}$ とする．TAN の構造学習では，次の尤度を最大にする構造 G^* を探索する．

$$G^* = \underset{G \in \mathcal{G}_{TAN}}{\operatorname{argmax}}\, p(D \mid G, \widehat{\Theta})$$

ここで，$\widehat{\Theta}$ は Θ の最尤推定量である．G^* を得るには，次の 5 つのステップを行えばよい．

1. 次の条件付き相互情報量 $I(X_i; X_h \mid X_0)$ を異なる二つの説明変数の組み $(X_i, X_h), i < h$ に対して計算する．

$$I(X_i; X_h \mid X_0) = \sum_{\substack{x_i \in \{1,\cdots,r_i\} \\ x_h \in \{1,\cdots,r_h\} \\ x_0 \in \{1,\cdots,r_0\}}} p(x_i, x_h, x_0) \log \frac{p(x_i, x_h \mid x_0)}{p(x_i \mid x_0) p(x_h \mid x_0)}$$

   条件付き相互情報量の計算に必要な確率 $p(x_i, x_h, x_0)$，$p(x_i, x_h \mid x_0)$，$p(x_i \mid x_0)$，$p(x_h \mid x_0)$ はそれぞれ次のように推定する．

   - $\hat{p}(x_i, x_h, x_0) = \dfrac{N_{x_i, x_h, x_0}}{N}$
   - $\hat{p}(x_i, x_h \mid x_0) = \dfrac{N_{x_i, x_h, x_0}}{N_{x_0}}$
   - $\hat{p}(x_i \mid x_0) = \dfrac{N_{x_i, x_0}}{N_{x_0}}$
   - $\hat{p}(x_h \mid x_0) = \dfrac{N_{x_h, x_0}}{N_{x_0}}$

   ここで，N_{x_i, x_h, x_0} は学習データにおいて $X_i = x_i, X_h = x_h, X_0 = X_0$ となる頻度であり，N_{x_i, x_0} は学習データにおいて $X_i = x_i, X_0 = X_0$ となる頻度である．

2. 各説明変数をノードとした完全無向グラフを生成し，各無向エッジ $(X_i, X_h), 1 \leq i < h \leq n$ に重み $I(X_i; X_h \mid X_0)$ を割り当てる．

3. 生成した重み付き完全グラフから，最大全域木を生成する．

4. 木のルートノードを一つ選び，そのルートノードから外側にエッジの方向をつけていく．

5. 目的変数から，構築された木構造の各説明変数に向けてエッジを加える．

演習問題 3

TAN のパラメータの最尤推定量 $\widehat{\theta}_{ijk}$ が上の値になることを確かめよ．
ヒント：束縛条件 $\sum_{k=1}^{r_i} \theta_{ijk} = 1$ を含んだラグランジュの未定乗数法を用いる．

演習問題 4

以下の”TAN.java”内の関数”getParameters”，”classification”，”setFrequencyTable”，”getConditionalMutualInformation”，”getMinimumSpanningTree”を実装し，TAN による分類プログラムを完成させよ．TAN.java がわかりにくい人は，TAN の分類プログラムを一から作成しても良い．
また，データセット”spam”,”sentiment”に対して TAN の分類精度を求め，Naive Bayes と比較・考察せよ．（TAN の学習時間が 1 分を超えるようなデータセットに関しては，分類精度は求めなくて良い．）
※TAN.java とデータセット”sentiment”は http://www.ai.lab.uec.ac.jp/実験/にある MICS_NB_TAN プロジェクト内にある．

ソースコード 2.2: TAN.java

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileReader;
4  import java.io.IOException;
5  import java.util.ArrayDeque;
6  import java.util.ArrayList;
7  import java.util.Comparator;
8  import java.util.HashMap;
9  import java.util.HashSet;
10 import java.util.PriorityQueue;
11 import java.util.Queue;
12 import java.util.Set;
13
14
15 public class TAN {
16   // Number of Categories
17   int NC = 2;
18   // Number of Variables
19   int NV;
20   // Leaning Data
21   ArrayList<int[]> LD;
22   // Test Data
23   ArrayList<int[]> TD;
24   //頻度表
25   double[] ft_c;
26   ArrayList<double[][]> ft_ic;
27   ArrayList<ArrayList<double[][]>> ft_ijc;
28   public TAN() throws IOException{
29     //データセット入力
30     String dataset = "banknote";
```

```
31    System.out.println("Dataset: "+dataset);
32    //学習データ読み込み
33    readLearningData("data/"+dataset+"/LD.csv");
34    //頻度表生成
35    setFrequencyTable();
36    //条件付き相互情報量
37    double[][] cmi;
38    cmi = getConditionalMutualInformation();
39    //TAN の構造学習
40    int[] str_tan = getMinimumSpanningTree(cmi);
41    //テストデータ読み込み
42    readTestingData("data/"+dataset+"/TD.csv");
43    //TAN のパラメータ学習
44    ArrayList<HashMap<Integer, Double>> parameters =
          getParameters(str_tan);
45    //学習したTAN で分類
46    classification(parameters, str_tan);
47  }
48
49  // 変数自身とその親の値を鍵としてその条件付き確率パラメータ
        を返すような
      HashMap を，各変数ごとに ArrayList で繋いだものを返す関数
50  private ArrayList<HashMap<Integer, Double>> getParameters(
        int[] str_tan){
51    ArrayList<HashMap<Integer, Double>> parameters = new
          ArrayList<>();
52
53
54
55
56
57    return parameters;
58  }
59
60  //パラメータを用いてテストデータの分類をし,その正答率を表示
        する.
61  private void classification(ArrayList<HashMap<Integer,
        Double>> parameters, int[] str_tan){
62    double num_correct_prediction = 0;
63    for(int d = 0; d < TD.size(); ++d){
64
65
66
67
68
69      //正答回数測定
```

```
70        if(predictedClass == TD.get(d)[NV - 1]){
71          num_correct_prediction++;
72        }
73      }
74      //正答率表示
75      System.out.println("Classification accuracy: "+
            num_correct_prediction / (double)TD.size());
76    }
77
78    //訓練データ読み込み
79    private void readLearningData(String DataBaseName) throws
          IOException{
80      BufferedReader br = new BufferedReader(new FileReader(new
            File(DataBaseName)));
81      ArrayList<String[]> tLD = new ArrayList<String[]>();
82      String line = br.readLine();
83      NV = line.split(",").length;
84      while(line != null){
85        String[] data = line.split(",").clone();
86        tLD.add(data);
87        line = br.readLine();
88      }
89      LD = new ArrayList<int[]>();
90      for(int i = 0; i < tLD.size(); ++i){
91        LD.add(new int[NV]);
92        for(int j = 0; j < NV; ++j){
93          LD.get(i)[j] = Integer.parseInt(tLD.get(i)[j]);
94        }
95      }
96      br.close();
97    }
98
99    //テストデータ読み込み
100   private void readTestingData(String DataBaseName) throws
          IOException{
101     BufferedReader br = new BufferedReader(new FileReader(new
            File(DataBaseName)));
102     ArrayList<String[]> tTD = new ArrayList<String[]>();
103     String line = br.readLine();
104     while(line != null){
105       String[] data = line.split(",").clone();
106       tTD.add(data);
107       line = br.readLine();
108     }
109     TD = new ArrayList<int[]>();
110     for(int i = 0; i < tTD.size(); ++i){
```

```
111       TD.add(new int[NV]);
112       for(int j = 0; j < NV; ++j){
113         TD.get(i)[j] = Integer.parseInt(tTD.get(i)[j]);
114       }
115     }
116     br.close();
117   }
118
119   //N_{x_0}, N_{ij},N_{ijk}をそれぞれ ft_c, ft_ic, ft_ijc に
          格納する関数
120   private void setFrequencyTable(){
121     //初期化
122     ft_c = new double[NC];
123     ft_ic = new ArrayList<>();
124     ft_ijc = new ArrayList<>();
125     for(int i = 0; i < NV - 1; ++i){
126       ft_ic.add(new double[NC][NC]);
127       ft_ijc.add(new ArrayList<>());
128       for(int empty = 0; empty <= i; ++empty){
129         ft_ijc.get(i).add(null);
130       }
131       for(int j = i + 1; j < NV - 1; ++j){
132         ft_ijc.get(i).add(new double[NC][NC][NC]);
133       }
134     }
135     //ft_c, ft_ic, ft_ijc を計算
136     for(int d = 0; d < LD.size(); ++d){
137
138
139
140
141     }
142   }
143
144   //ft_c, ft_ic, ft_ijc を用いて条件付き相互情報量を求める
145   private double[][] getConditionalMutualInformation(){
146     double[][] cmi = new double[NV - 1][NV - 1];
147
148
149
150     return cmi;
151   }
152
153   // 条件付き相互情報量を重みとして最小全域木を求める関数
154   // 説明変数の構成する木構造のルートノードはx_0 とする.
155   // 各変数の親変数 (目的変数以外)を格納した配列を返す.
```

```
156   private int[] getMinimumSpanningTree(double[][] cmi){
157
158
159
160   }
161
162   public static void main(String[] args) throws IOException {
163     new TAN();
164   }
165 }
```

これまで紹介した Naive Bayes と TAN では，パラメータを最尤法で推定した．しかし，よく知られているように，ベイズ推定はより強力である．以下で紹介しよう．

第3章　ディリクレモデル

ベイズアプローチに従い，パラメータの事前分布を考えることにしよう．事前分布を考える場合，様々な考え方があるが，最も合理的であると考えられるのは，事前分布と事後分布の分布形が同一になるような事前分布，すなわち，自然共益事前分布の導入であろう．尤度は多項分布に従うので，その自然共益事前分布であるディリクレ分布が事前分布としてよく用いられる．ここで，G を Naive Bayes または TAN の構造とし，目的変数のパラメータ集合を $\Theta_0 = \{\theta_{x_0}\}, (x_0 = 1, \cdots, r_0)$ とし，説明変数 X_i の親変数集合が j 番目パターンをとる時のパラメータ集合を $\Theta_{ij} = \{\theta_{ijk}\}, (k = 1, \cdots, r_i)$ とすると，ディリクレ分布 $p(\Theta_0 \mid G)$，$p(\Theta_{ij} \mid G)$ はそれぞれ次のように表せる．

$$p(\Theta_0 \mid G) = \frac{\Gamma\left(\sum_{x_0=1}^{r_0} \alpha_{x_0}\right)}{\prod_{x_0=1}^{r_0} \Gamma(\alpha_{x_0})} \prod_{x_0=1}^{r_0} \theta_{x_0}^{\alpha_{x_0}-1},$$

$$p(\Theta_{ij} \mid G) = \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}$$

ここで，α_{x_0} は N_{x_0} に，α_{ijk} は N_{ijk} に対応する事前の知識を表現する擬似サンプルとしてのハイパーパラメータを示す．

事後分布は，事前分布を尤度に掛け合わせることにより得ることができる．先に求められた尤度とディレクレ分布を掛け合わせるとそれぞれ以下のような事後分布を得ることができる．

$$\begin{aligned} p(D, \Theta_0 \mid G) &= \frac{\Gamma\left(\sum_{x_0=1}^{r_0} \alpha_{x_0}\right)}{\prod_{x_0=1}^{r_0} \Gamma\left(\alpha_{x_0}\right)} \prod_{x_0=1}^{r_0} \theta_{x_0}^{\alpha_{x_0}+N_{x_0}-1} \qquad (3.1) \\ &\propto \prod_{x_0=1}^{r_0} \theta_{x_0}^{\alpha_{x_0}+N_{x_0}-1} \end{aligned}$$

$$\begin{aligned} p(D, \Theta_{ij} \mid G) &= \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}+N_{ijk}-1} \qquad (3.2) \\ &\propto \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}+N_{ijk}-1} \end{aligned}$$

これらの事後分布を最大にする MAP(Maximum A Posteriori) 推定値は，以下のとおりである．

$$\widehat{\theta}_{x_0} = \frac{\alpha_{x_0} + N_{x_0} - 1}{\alpha + N - r_0} \qquad (3.3)$$

$$\widehat{\theta}_{ijk} = \frac{\alpha_{ijk} + N_{ijk} - 1}{\alpha_{ij} + N_{ij} - r_i} \tag{3.4}$$

ここで，$\alpha = \sum_{j=1}^{r_0} \alpha_{x_0}$，$\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$.

MAP 推定値では，全てのハイパーパラメータを $\alpha_{x_0} = 1$，$\alpha_{ijk} = 1$ として一様分布に設定すると最尤推定値 (Maximum Likelihood estimator) に一致する．

しかし，ベイズ統計学では，MAP 推定値よりも事後分布の期待値である EAP(Expected A Posteriori) 推定値のほうが頑健で予測効率がよいことが知られている．ディリクレ分布，式 (3.1)，(3.2) の期待値は

$$\widehat{\theta}_{x_0} = \frac{\alpha_{x_0} + N_{x_0}}{\alpha + N} \tag{3.5}$$

$$\widehat{\theta}_{ijk} = \frac{\alpha_{ijk} + N_{ijk}}{\alpha_{ij} + N_{ij}} \tag{3.6}$$

となる．ベイズ学習では，EAP 推定値が最も一般的に用いられる．

演習問題 5

NB と TAN のパラメータを最尤推定量と EAP 推定量のそれぞれで推定した時の，データセット”spam”,”sentiment”の分類精度を求めよ．また，その結果から最尤推定と EAP 推定の違いを比較・考察せよ．

第4章　ベイジアン・ネットワーク

本章では，ベイジアン・ネットワークの定義を与える．そのために，まず，最小 I-map と d 分離の概念を紹介しよう．

4.1　条件付き独立構造のグラフ表現

グラフィカル・モデルでは，グラフ上で以下のように条件付き独立性を表現する．

定義 22 X, Y, Z が無向グラフ G の互いに排他なノード集合であるとする．もし，X と Y の各ノード間のすべての路が Z の少なくとも一つのノードを含んでいるとき，Z は X と Y を分離する，といい，$I(X,Y \mid Z)_G$ と書く．これは，グラフ上で条件付き独立性を表現する．一方，真の条件付き独立性，すなわち，X と Y が Z を所与として条件付き独立であるとき，$I(X,Y \mid Z)_M$ と書く．

上の条件付き独立性について，下のような様々な性質が知られている (Lauritzen(1974), Dawid(1979,1980)).

1. 対象性 (Symmetry):

$$I(X,Y \mid Z) \Leftrightarrow I(Y,X \mid Z)$$

2. 分離性 (Decomposition):

$$I(X,Y \cup W \mid Z) \Rightarrow I(X,Y \mid Z) \text{ and } I(X,W \mid Z)$$

3. 弱結合性 (Weak Union)

$$I(X,Y \cup W \mid Z) \Rightarrow I(X,W \mid Z \cup Y) \text{ and } I(X,Y \mid Z \cup W)$$

4. 縮約性 (Contraction)

$$I(X,W \mid Z \cup Y) \text{ and } I(X,Y \mid Z) \Rightarrow I(X,Y \cup W \mid Z)$$

5. 交差性 (Intersection)

$$I(X,W \mid Z \cup Y) \text{ and } I(X,Y \mid Z \cup W) \Rightarrow I(X,Y \cup W \mid Z)$$

6. 強結合性 (Strong Union)

$$I(X,Y \mid Z) \Rightarrow I(X,Y \mid Z \cup W)$$

7. 強推移性 (Strong Transitivity)

$$I(X,Y \mid Z) \Rightarrow I(X,A \mid Z) \text{ or } I(A,Y \mid Z)$$

ただし,$A \notin X \cup Y \cup Z$

8. 弱推移性 (Weak Transitivity)

$$I(X,Y \mid Z) \text{ and } I(X,Y \mid Z \cup A) \Rightarrow I(X,A \mid Z) \text{ or } I(A,Y \mid Z)$$

9. 弦可能性 (Chordality)

$$I(A,C \mid B \cup D) \text{ and } I(B,D \mid A \cup C) \Rightarrow I(A,C \mid B) \text{ or } I(A,C \mid D)$$

定義 23 グラフ G は，ノードに対応する変数間すべての真の条件付き独立性がグラフでの表現に一致し，さらにその逆が成り立つとき，G をパーフェクト・マップ *(Perfect Map)* といい，*P-map* と書く．

$$I(X,Y \mid Z)_M \Leftrightarrow I(X,Y \mid Z)_G$$

しかし，すべての確率モデルに対応するグラフが存在するわけではない．例えば，４つの変数 X_1, X_2, Y_1, Y_2 が以下の真の条件付き独立性を持っているとする．

$$I(X_1, X_2 \mid Y_1, Y_2)_M$$
$$I(X_2, X_1 \mid Y_1, Y_2)_M$$
$$I(Y_1, Y_2 \mid X_1, X_2)_M$$
$$I(Y_2, Y_1 \mid X_1, X_2)_M$$

しかし，この条件付き独立性を表現できるグラフ表現は存在しない．そこで，グラフ表現できる条件付き独立性のみを扱うことにし，次の I-Map が導入される．

定義 24 グラフ G は，グラフでの条件付き独立性のすべての表現が真の条件付き独立性に一致しているとき，G をインデペンデント・マップ *(Independent Map)* といい，*I-map* と書く．

$$I(X,Y \mid Z)_G \Rightarrow I(X,Y \mid Z)_M$$

I-map の定義では，完全グラフを仮定するとグラフ上に $I(X,Y \mid Z)_G$ が存在しないので，どんな場合でも I-map を満たしてしまう．そこで、以下の最小 I-map が重要となる．

定義 25 グラフ G が *I-map* で，かつ，一つでもエッジを取り除くとそれが *I-map* でなくなってしまうとき，グラフ G は最小 *I-map(minimal I-map)* と呼ばれる．

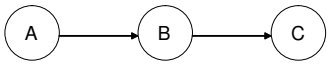


図 4.1: 逐次結合（Serial connections）

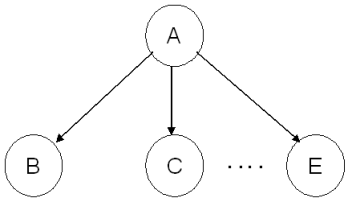


図 4.2: 分岐結合（Diverging connections）

4.2 d 分離

ベイジアン・ネットワークの定義では,「d 分離」の概念が基底をなす．d 分離とは，有向グラフにおいて，以下の三つの結合の中で定義することができる．

逐次結合（Serial connections）

図 3.1 の構造を考える．A は B に影響を与え，B は C に影響を持つ．明らかに，A についての証拠は，B の確からしさに影響を与え，さらに C の確からしさに影響を与える．同様に，C についての証拠は B を通じて A の確からしさに影響を持つ．一方，B の状態がわかってしまったら，通路は完全にブロックされてしまう，A と C は独立になってしまう．このようなとき，A と C は,「**B を所与として d 分離（d-separtion）である**」と呼び，一つの変数の状態がわかることを,「**その変数がインスタンス化された**」（it is instantiated）と呼ぶ．以上をまとめると，逐次結合では，結合部の変数の状態がわからない限り，証拠は変数間を伝播する．

分岐結合（Diverging connections）

図 3.2 の構造は，分岐結合と呼ばれる．A の状態がわからない限り，A の全ての子の間で証拠が伝播される．この場合，B,C, ··,E は,「**A を所与として d 分離である**」という．

例 11 例えば，図 *3.3* は成人についての性別（男，女）と髪の長さ（短い，長い），身長（$> 168cm, < 168cm$）との因果ネットワークを示している．もし，人の性別を知らない場合，髪の毛の長さを知ることによって性別についての

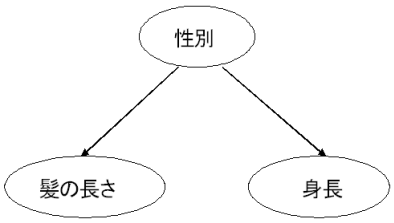


図 4.3: 成人についての性別と髪の長さ，身長との因果ネットワーク

情報を得ることができるし，それが身長への推論にも役立つであろう．一方，一旦その人が男であることを知ってしまったら，髪の長さは身長を知るための新しい手がかりにはならないであろう．

合流結合 (Converging connections)

図 3.4 の構造は少し注意が必要である．例えば，図中の変数 A について何の情報も持っていないときには，どれかの変数について証拠を得ても他の変数には全く影響を与えることはない．しかし，一旦，A についての証拠を得てしまうと，どのような親の証拠でも他の親変数の確からしさに影響を与える．これは,「**説明効果（explaninng away effect）**」と呼ばれる．例えば，図 3.4 の変数 A について a が起こったとき，親 B の事象 b と親 C の事象 c は共に a を起こす原因であるとする．このとき，c が起こったという情報は b が起こる確からしさを減少させる．また，c が起こらなかったという情報は b が起こる情報を増加させるのである．結果として，結合部の変数もしくはその子孫の証拠が得られたときのみ，合流結合における証拠は伝播される．図 3.4 の場合も，変数 B,C, ··,E は，A もしくは A の子孫についての証拠を得ない場合（**開いている (Opening) 場合**）,「**A を介して d 分離である**」と呼ぶ.

変数についての証拠はその状態の確からしさの記述である．もし，その変数の値が観測されていた場合,「変数がインスタンス化されている」と呼び，その値を「エビデンス」(evidence)，特に値が知られている場合を「**ハード・エビデンス（Hard evidence）**」と呼ぶ．例えば，性別変数について，男であることがわかったなら，それはハード・エビデンスである．それ以外であれば，そのエビデンスを「**ソフト・エビデンス (soft evidence)**」と呼ぶ．例えば，性別変数について，男である確率が 0.7 であることがわかった場合，それはソフト・エビデンスである．逐次結合と分岐結合でのブロックのため，または合流結合において開いているためには，ハード・エビデンスが必要である．

例 12 図 *3.5* は食べ過ぎおよび風邪と吐き気，顔色との因果ネットワークである．今，吐き気と顔色について何の情報もなければその人が食べ過ぎかど

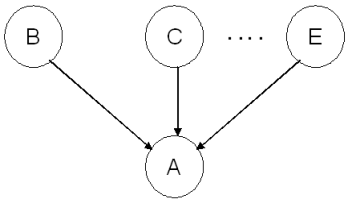


図 4.4: 合流結合 (Converging connections)

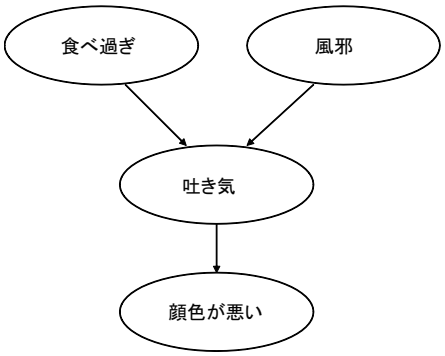


図 4.5: 顔色の因果ネットワーク

うかはその人が風邪を引いているかどうかには何も影響を与えない．すなわち，独立である．しかし，もし，その人の顔色が青ざめていることがわかったら，その人が食べ過ぎかどうかの情報はその人が風邪を引いているかどうかどうかの確からしさに影響を与えるのである．

4.2.1 d 分離

定義 26 因果ネットワークにおける二つの変数 A と B は，A と B の全てのパスに存在する以下のような変数 V*(A と B を分ける)* があるとき，**d 分離**（**d-separate**）である．

- 逐次結合もしくは分岐結合で V がインスタンス化されているとき，又は
- 合流結合で V もしくは V の子孫がインスタンス化されていないとき

A と B が d 分離でないとき，d 結合（*d-connection*）と呼ぶ．

図 3.6 は B と M がインスタンス化された因果ネットワークの例である．A についてのエビデンスが入力されたとき，それは D に伝播される．B はブロックされているのでそのエビデンスは B を通じて E に伝播されることはない．しかし，H と K には通じているので K の子の M にエビデンスが与えら

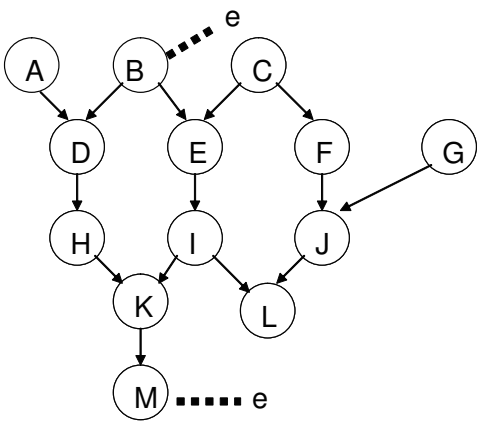


図 4.6: B と M がインスタンス化された因果ネットワーク

れているために，H からのエビデンスは I へ伝播され，さらに E,C,F,J,L へと伝播される．つまり，経路 A-D-H-K-I-E-C-F-J-L は d 結合の経路である．

ノート 1 A と B が d 結合であっても，A の確からしさの変化が必ずしも B の確からしさに影響を与えるとは限らない．値の与えようによっては $P(A \mid B)P(B) = P(A)P(B)$ となってしまっているかもしれない．もし，このような場合があれば,「A と B は構造的に独立である *(structually independet)*」という．

定理 13 *Jensen and Nielsen (2007)*
もし，A と B が d 分離であれば，A の確からしさにおける変化は B の確からしさに全く影響を与えない．

定理 14　*Darwiche (2009)*
ある因果ネットワークにおいて，二つのノード集合 X と Y が，ノード集合 Z によって d 分離されることは，以下の枝刈り *(Pruning)* によって得られる新しい因果ネットワークにおいて，X と Y が分離されていることと同値である．

- $X \cup Y \cup Z$ に属していない全ての葉ノード *(leaf nodes)* を削除する．
- Z のノードから引かれたアークを全て削除する．

例 13 図 *3.7* における因果ネットワークで，$X = \{B, D\}$,$Z = \{C, H, I, K\}$, $Y = \{M, N\}$ とする．$X \cup Y \cup Z = \{B, C, D, H, I, K, M, N\}$ であり，それ以外のノードを削除する．Z の要素からのアークを全て削除すると，X と Y は完全に分離される．すなわち，X と Y がノード集合 Z を所与として d 分離されている．

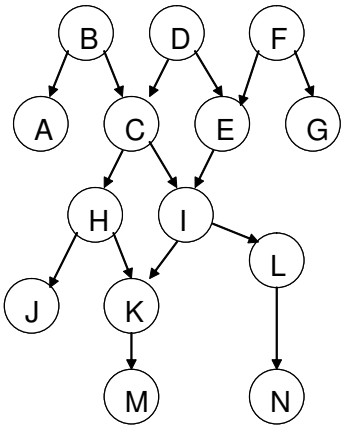


図 4.7: 因果ネットワークの例

定義 27 変数 A の**マルコフ・ブランケット (Markov Blanket)** とは，A の親集合，子集合，A と子を共有している変数集合の和集合より成り立つ

ノート 2 A についてのマルコフ・ブランケットがすべてインスタンス化されている場合，A はネットワーク中の残りの変数すべてと d 分離である．

例 14 図 *3.7* において，I についてのマルコフ・ブランケットは *(C,E,H,K,L)* となる．ただし，I の近傍の変数のみがインスタンス化されている場合，J は I とは d 分離でないことに注意してほしい．なぜならば，この場合，I はインスタンス化されないのであるが，K を所与として合流結合である親 H は I から影響を受けるのである．

定義 28 X, Y, Z がグラフ G の互いに排他なノード集合であるとする．X と Y が Z によって d 分離されるとき，$I(X,Y \mid Z)_d$ と書く．

定理 15 $I(X,Y \mid Z)_d$ と $I(X,Y \mid Z)_G$ は同値である．すなわち，

$$I(X,Y \mid Z)_d \Leftrightarrow I(X,Y \mid Z)_G$$

このように，d 分離はグラフ上で表現するには都合のよい概念であり，この仮定がベイジアン・ネットワークのモデルそのものである．真の条件付き独立性全ては表現できておらず，最小 I-map を仮定していることになる．

また，d 分離は以下のようにも定義することができる．

定義 29 X, Y, Z がグラフ G の互いに排他なノード集合であるとする．X, Y, Z を含む最小のアンセスタル集合のモラル・グラフで，Z が X と Y を分離するとき， Z は X と Y を d 分離する．

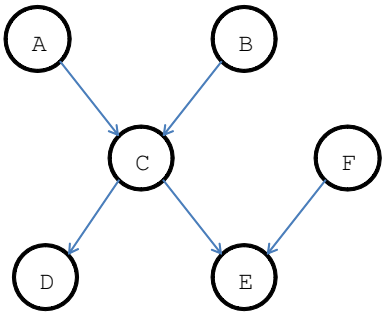


図 4.8: 有向グラフ

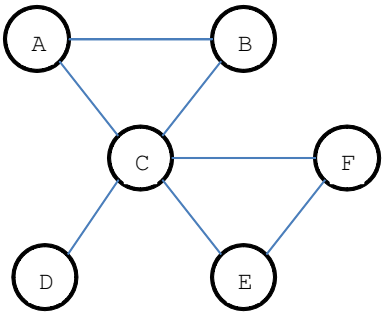


図 4.9: 図 3.8 のモラル・グラフ

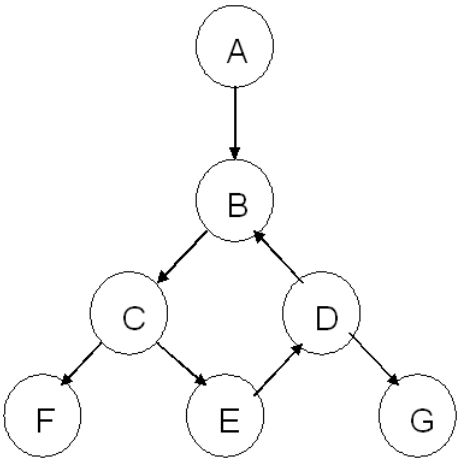


図 4.10: 循環フィードバックを含む DAG. ベイジアン・ネットワークでは許されない.

例 15 図 *3.8* における有向グラフを考える．これを無向化し，モラル化したグラフは図 *3.9* である．例えば，E は C と F を d 分離しない，C は D と F を d 分離する，などがわかる．

4.3　ベイジアン・ネットワーク・モデル

4.3.1　定義

定義 30 N 個の変数集合 $x = \{x_1, x_2, \ldots, x_N\}$ を持つベイジアン・ネットワークは，(G, Θ) で表現される．

- G は x に対応するノード集合によって構成される非循環有向グラフ *(Directed acyclic graph; DAG)*，ネットワーク構造と呼ばれる．

- Θ は，G の各アークに対応する条件付き確率パラメータ集合 $\{p(x_i \mid \Pi_i, G)\}, (i = 1, \cdots, N)$ である． ただし，Π_i は変数 x_i の親変数集合を示している．

ここで，DAG とは $A_1 \to \cdots \to A_n$, s.t. $A_1 = A_n$ となるような有向経路がないことである．逆に循環有向グラフは図 3.10 のように，循環 $B \to C \to E \to D \to B$ を含んでいる有向グラフを意味する．ベイジアン・ネットワークの定義では、因果の定義やエッジが因果の強さに関係することなどは言及されていない．

そのかわりに、構造に含まれる d 分離の性質が成り立つことが必要となるのである．これは A と B がエビデンス e を得たときに d 分離となる場合、ベイジアン・ネットワークで用いられる確率計算は $P(A \mid e) = P(A \mid B, e)$ となることを意味している．

ベイジアン・ネットワークの構造を構築するとき、必ずしも因果の方向に向かうエッジが必要なわけではない．そのかわり、その d 分離の性質を確認することが必要となる．

$x = \{x_1, x_2, \ldots, x_N\}$ を変数集合とする．もし、同時確率 $p(x) = p(x_1, x_2, \ldots, x_N)$ を全て持っているのであれば、$p(x_i)$ や $p(x_i \mid e)$ を計算することができる．しかし、$p(x)$ は変数数によって指数的に大きくなってしまう．故に我々はよりコンパクトな $p(x)$ の表現、必要なら $p(x)$ が計算される情報の蓄積方法を探さねばならない．x についてのベイジアン・ネットワークとはそのような表現である．

定理 16 変数集合 $x = \{x_1, x_2, \ldots, x_N\}$ を持つベイジアン・ネットワークの同時確率分布 $p(x)$ は以下で示される．

$$p(x \mid G) = \prod_i P(x_i \mid \Pi_i, G), \tag{4.1}$$

ここで、G は最小 *I-map* を示している．

この定理はベイジアンネットワークの基礎となるので, 証明を示しておこう．

証明 1 x を n 個の変数を持つ *DAG* を考えよう．ネットワークは非循環なので, 子のいない変数 A が少なくとも一つは存在する（図 *3.11)*. 次に,A をネットワークから除去することを考える. チェーンルールより, $p(x\backslash\{A\})$ は $p(A \mid \Pi_A)$ を除くすべての条件付き確率の積である.

$$p(x) = p(A \mid x\backslash\{A\}) \cdot p(x\backslash\{A\}).$$

A は Π_A を所与として $x\backslash(\{A\} \cup \Pi_A)$ と d 分離であるので、

$$p(x) = p(A \mid x\backslash\{A\})p(x\backslash\{A\}) = p(A \mid \Pi_A) \cdot p(x\backslash\{A\}).$$

$p(x\backslash\{A\})$ についても同様に変形でき, 同時確率分布は親ノード変数を所与とする全ての条件付き確率の積, 式 *(3.1)* で示されることが証明される．すなわち, ベイジアン・ネットワークとは，非循環性と d 分離の仮定のみによって導かれる現在考えられる最も自然な離散モデルであり，現在の様々なモデルの中でも, 最も表現力と予測力を持つモデルである．

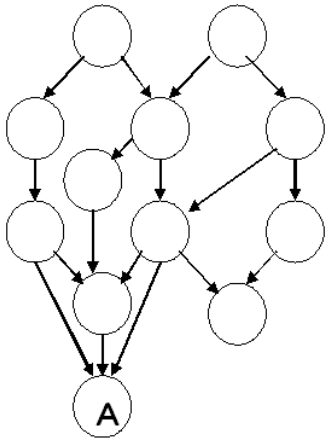


図 4.11: n 変数を持つ DAG

4.3.2　ベイジアン・ネットワークと d 分離

先の定理の証明では d 分離の性質を用いた．そのために d 分離は式 (3.1) の中に埋め込まれていることになる．別の言い方をすれば、もし C が集合 V を所与として A と d 分離であるとき, 式 (3.1) から $P(C \mid A,V) = P(C \mid V)$ を推論できる．以下にそれを示す.

[逐次結合の場合] A が B を通じて C に逐次結合している場合を考えよう．このとき、$P(C \mid B,A) = P(C \mid B)$ を示せばよい．チェーンルールより、

$$P(A,B,C) = P(A)P(B \mid A)P(C \mid B) = P(A,B)P(C \mid B)$$

よって

$$P(C \mid B,A) = \frac{P(A,B,C)}{P(A,B)} = P(C \mid B)$$

となる.

[分岐結合の場合] A の独立な二つの子が B, C であるとする．このとき, $P(C \mid A,B) = P(C \mid A)$ を示せばよい．チェーンルールより、

$$P(A,B,C) = P(A)P(B \mid A)P(C \mid A) = P(A,B)P(C \mid A)$$

よって

$$P(C \mid A, B) = \frac{P(A, B, C)}{P(A, B)} = P(C \mid A)$$

となる．

[合流結合の場合] A と B を C の親としよう．今、$P(A \mid B) = P(A)$ を示せばよい．チェーンルールより

$$P(A, B, C) = P(A)P(B)P(C \mid B, A)$$

C について周辺化し,

$$P(A, B) = \sum_C P(A)P(B)P(C \mid B, A)$$

ここで,$\sum_C$ は $P(C \mid B, A)$ にのみ掛かっているので,

$$\sum_C P(A)P(B)P(C \mid B, A) = P(A)P(B) \sum_C P(C \mid B, A)$$

条件付き確率表で列の和は 1 になるので、$\sum_C P(C \mid B, A)$ は 1 となる．したがって $P(A, B) = P(A)P(B)$ となるのである．

4.3.3 ベイジアン・ネットワークの実際の表現と推論

具体的には，ベイジアン・ネットワークは，DAG で示されるネットワーク構造 G と CPT(Conditional Probabities Tables) と呼ばれる条件付き確率表によって表現される．例えば，図 3.12 は芝生が濡れていたとき，それがスプリンクラーによるものか雨によるものかを推論するベイジアン・ネットワークの構造と CPT である．

今，状態が真のとき 1，偽のときに 0 をとる確率変数 $\{A, B, C, D, E\}$ を導入しよう．ベイジアン・ネットワークの同時確率分布は，式 (3.1) で示されるので，例えば，$p(A = 1, B = 1, C = 1, D = 1, E = 1 \mid G)$ は

$$\begin{aligned} & p(A = 1, B = 1, C = 1, D = 1, E = 1 \quad \mid G) \\ = \;& p(A = 1)p(B = 1 \mid A = 1)p(C = 1 \mid A = 1) \\ & p(D = 1 \mid B = 1, C = 1)p(E = 1 \mid C = 1) \\ = \;& 0.6 \times 0.2 \times 0.8 \times 0.95 \times 0.7 = 0.06384 \end{aligned}$$

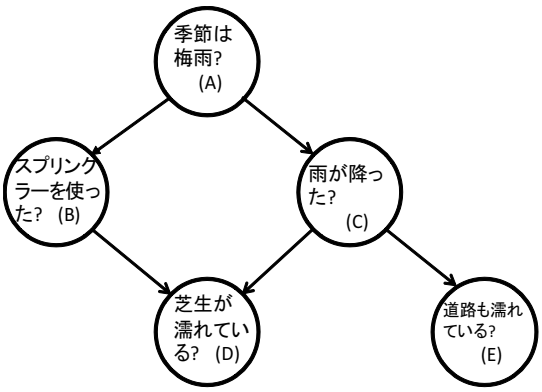
同様に図 3.12 のすべての変数の状態について同時確率を計算し，得られた同時確率分布表 (JPDT:Joint Probability Distribution Table) は表 3.1 のとおりである．

実装上のベイジアン・ネットワークでは，このように CPT や JPDT を用いて計算する．そのために，CPT や JPDT を区別せず，関数 φ として定義する．

tbh

表 4.1: 図 3.12 の同時確率分布表:JPDT

A	B	C	D	E	$p(A,B,C,D,E)$
1	1	1	1	1	0.06384
1	1	1	1	0	0.02736
1	1	1	0	1	0.00336
1	1	1	0	0	0.00144
1	1	0	1	1	0.0
1	1	0	1	0	0.02160
1	1	0	0	1	0.0
1	1	0	0	0	0.00240
1	0	1	1	1	0.21504
1	0	1	1	0	0.09216
1	0	1	0	1	0.05376
1	0	1	0	0	0.02304
1	0	0	1	1	0.0
1	0	0	1	0	0.0
1	0	0	0	1	0.0
1	0	0	0	0	0.09600
0	1	1	1	1	0.01995
0	1	1	1	0	0.00855
0	1	1	0	1	0.00105
0	1	1	0	0	0.00045
0	1	0	1	1	0.0
0	1	0	1	0	0.24300
0	1	0	0	1	0.0
0	1	0	0	0	0.02700
0	0	1	1	1	0.00560
0	0	1	1	0	0.00240
0	0	1	0	1	0.00140
0	0	1	0	0	0.00060
0	0	0	1	1	0.0
0	0	0	1	0	0.0
0	0	0	0	1	0.0
0	0	0	0	0	0.0900



A	p(A)
真	0.6
偽	0.4

A	B	p(B|A)
真	真	0.2
真	偽	0.8
偽	真	0.75
偽	偽	0.25

A	C	p(C|A)
真	真	0.8
真	偽	0.2
偽	真	0.1
偽	偽	0.9

B	C	D	p(D|B,C)
真	真	真	0.95
真	真	偽	0.05
真	偽	真	0.9
真	偽	偽	0.1
偽	真	真	0.8
偽	真	偽	0.2
偽	偽	真	0.0
偽	偽	偽	1.0

C	E	p(E|C)
真	真	0.7
真	偽	0.3
偽	真	0.0
偽	偽	1.0

図 4.12: ベイジアン・ネットワークの CPT

定義 31 変数集合 $\mathbf{X}$ のファクター *(factor)*φ *(Jensen* らはポテンシャル関数 *(potential function)* と呼ぶ*)* とは，変数集合 $\mathbf{X}$ の各値 $\mathbf{x}$ を非負値に写像させる関数であり，$\varphi(\mathbf{x})$ と書く．

実際には，各変数の状態確率に興味があるので，以下のように N 個の変数を持つ同時確率分布 $p(x_1, x_2, \ldots, x_N \mid G)$ で対象となる変数 x_i 以外の変数を周辺化して $p(x_i \mid G)$ を求めればよい．

$$p(x_i \mid G) = \sum_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_N} p(x_1, x_2, \ldots, x_N \mid G) \tag{4.2}$$

具体的には，アルゴリズム 6 により変数消去が行われ，ベイジアン・ネットワークより周辺確率を求めることが可能となる．

アルゴリズム 1　周辺事前確率のための変数消去アルゴリズム

- **Input:** ベイジアン・ネットワーク $\{G, \Theta\}$, ベイジアン・ネットワークでのクエリ *(query)* 変数集合 Q
- **Output:** 周辺確率 $p(Q \mid G)$

1. $\mathcal{S} \leftarrow$ *CPT* の値

2. **for** *i=1 to N* **do**

3. $\varphi \leftarrow \prod_k \varphi_k$, ここで φ_k は Q に含まれないノード i に関係する（を含む）$\mathcal{S}$ に属する条件付き確率

4. $\varphi_i \leftarrow \sum_i \varphi$

5. $\mathcal{S}$ の全ての φ_k を φ_i によって置き換える

6. **end for**

7. **return** $\prod_{\varphi \in \mathcal{S}} \varphi$

アルゴリズム 1 では，ベイジアン・ネットワーク構造と CPT を $\mathcal{S}$ に読み込み，ノード番号順にノード i が Q 以外の変数であれば，i 番目の変数を含むファクターを全て積算して，i 番目の変数について足し合わせる．これを繰り返し，クエリ (query) 変数集合 Q の周辺確率を得るというものである．

例 16 例えば，図 *3.12* について周辺確率 $p(E = 1 \mid G)$ をアルゴリズム *6* に従い計算すると以下のようになる．

$$\sum_D \sum_C \sum_B p(E \mid C) p(D \mid B, C) \sum_A p(A) p(B \mid A) p(C \mid A) = 0.364$$

ここでの変数の周辺化は，ベイジアン・ネットワークのいくつかの変数がインスタンス化される（エビデンスを得る）前の事前確率分布について行われるものであり，得られた各変数の周辺確率を周辺事前確率 (marginal prior) と呼び，この操作を事前分布周辺化 (prior marginals) と呼ぶ．それに対して，ベイジアン・ネットワークでいくつかの変数がインスタンス化（エビデンスを得る）された場合の各変数の周辺確率を周辺事後確率 (marginal posterior) と呼び，この操作を事後分布周辺化 (posterior marginals) と呼ぶ．

エビデンス $\mathbf{e}$ を所与とした周辺事後確率を計算する場合（エビデンスを得た場合の変数消去の場合），まず同時周辺確率 (joint marginals)$p(Q, \mathbf{e} \mid G)$ を計算する．そのために，エビデンスに一致しないファクターの値を 0 に変換するように，ファクターを以下のように再定義する．

定義 32 エビデンス $\mathbf{e}$ を所与としたときのファクター $\varphi^e(\mathbf{x})$ は以下のように定義される．

$$\varphi^e(\mathbf{x}) = \begin{cases} \varphi(\mathbf{x}) & : \ \mathbf{x} \text{ が } \mathbf{e} \text{ に一致しているとき} \\ 0 & : \ \text{上記以外} \end{cases}$$

さらに，この変換について以下の分配法則が成り立つ．

定理 17 φ_1 と φ_2 が二つの異なるファクターであり, エビデンス $\mathbf{e}$ を得たとき,

$$(\varphi_1 \varphi_2)^e = \varphi_1^e \varphi_2^e$$

が成り立つ．

これらの性質を用いると，エビデンス $\mathbf{e}$ を得たときの，周辺事後確率を求めるための変数消去アルゴリズムはアルゴリズム 6 の CPT を S に組み入れるのを φ^e 変換を行うことによりアルゴリズム 7 が得られる．

アルゴリズム 2　周辺事後確率のための変数消去アルゴリズム

- **Input:** ベイジアン・ネットワーク $\{G, \Theta\}$, ベイジアン・ネットワークのクエリ変数集合 Q，エビデンス $\mathbf{e}$
- **Output:** $p(Q, \mathbf{e} \mid G)$

1. $\mathcal{S} \leftarrow \varphi^e \leftarrow \varphi$

2. **for** *i=1 to N* **do**

3. $\varphi \leftarrow \prod_k \varphi_k$, ここで φ_k はノード i に関係する（を含む）$\mathcal{S}$ に属する φ^e

4. $\varphi_i \leftarrow \sum_i \varphi$

5. $\mathcal{S}$ の全ての φ_k を φ_i によって置き換える

6. **end for**

7. **return** $\prod_{\varphi \in \mathcal{S}} \varphi$

例 17 今，エビデンス $\mathbf{e} = \{A = 1, B = 0\}$*(*真のとき 1, 偽のとき 0*)* とし，$Q = \{D, E\}$ とする. 図 *3.12* の *CPT* より，φ^e を求めると図 *3.13* のファクター表が得られる．アルゴリズム *2* に従い，図 *3.13* のファクター表を用いて $p(D, E, \mathbf{e} \mid G)$ を求めると以下の式に従うことになる．

$$
\begin{aligned}
p(D, E, \mathbf{e} \mid G) &= \sum_A \sum_B \sum_C \varphi^e(E \mid C)\varphi^e(D \mid B, C)\varphi^e(A)\varphi^e(B \mid A)\varphi^e(C \mid A) \\
&= \sum_{C=\{0,1\}} \varphi^e(E \mid C)\varphi^e(D \mid B, C)\varphi^e(C \mid A)
\end{aligned}
$$

を得る．この数式に従い，図 *3.13* のファクター表を用いて各周辺事後確率の

A	$\varphi^e(A)$
真	1.0
偽	0.0

A	B	$\varphi^e(B\|A)$
真	真	0.0
真	偽	1.0
偽	真	0.0
偽	偽	0.0

A	C	$\varphi^e(C\|A)$
真	真	0.8
真	偽	0.2
偽	真	0.0
偽	偽	0.0

B	C	D	$\varphi^e(D\|B,C)$
真	真	真	0.0
真	真	偽	0.0
真	偽	真	0.0
真	偽	偽	0.0
偽	真	真	0.8
偽	真	偽	0.2
偽	偽	真	0.0
偽	偽	偽	1.0

C	E	$\varphi^e(E\|C)$
真	真	0.7
真	偽	0.3
偽	真	0.0
偽	偽	1.0

図 4.13: 図 3.12 の CPT についてエビデンス $\mathbf{e} = \{A = 1, B = 0\}$ を得たときの φ^e

ファクターは以下のように求められる．

$$
\begin{aligned}
\varphi(D=1,E=1,\mathbf{e}\mid G) &= \varphi^e(E=1\mid C=0)\varphi^e(D=1\mid B=0,C=0)\varphi^e(C=0\mid A=1)\\
&\quad +\varphi^e(E=1\mid C=1)\varphi^e(D=1\mid B=0,C=1)\varphi^e(C=1\mid A=1)\\
&= (0.0\times 0.0\times 0.2)+(0.7\times 0.8\times 0.8)=0.448\\
\varphi(D=1,E=0,\mathbf{e}\mid G) &= \varphi^e(E=0\mid C=0)\varphi^e(D=1\mid B=0,C=0)\varphi^e(C=0\mid A=1)\\
&\quad +\varphi^e(E=0\mid C=1)\varphi^e(D=1\mid B=0,C=1)\varphi^e(C=1\mid A=1)\\
&= (0.0\times 0.0\times 0.2)+(0.7\times 0.8\times 0.8)=0.192\\
\varphi(D=0,E=1,\mathbf{e}\mid G) &= \varphi^e(E=1\mid C=0)\varphi^e(D=0\mid B=0,C=0)\varphi^e(C=0\mid A=1)\\
&\quad +\varphi^e(E=1\mid C=1)\varphi^e(D=0\mid B=0,C=1)\varphi^e(C=1\mid A=1)\\
&= (0.0\times 1.0\times 0.2)+(0.7\times 0.2\times 0.8)=0.112\\
\varphi(D=0,E=0,\mathbf{e}\mid G) &= \varphi^e(E=0\mid C=0)\varphi^e(D=0\mid B=0,C=0)\varphi^e(C=0\mid A=1)\\
&\quad +\varphi^e(E=0\mid C=1)\varphi^e(D=0\mid B=0,C=1)\varphi^e(C=1\mid A=1)\\
&= (1.0\times 1.0\times 0.2)+(0.3\times 0.2\times 0.8)=0.248
\end{aligned}
$$

結果として，エビデンス $\mathbf{e}$ を所与とした各周辺事後確率は，今,$p(\mathbf{e}) = 1.0$ な

ので各ファクターを $p(\mathbf{e}) = 1.0$ で除しても変化せず，

$$\begin{aligned} p(D = 1, E = 1 \mid G, \mathbf{e}) &= 0.448 \\ p(D = 1, E = 0 \mid G, \mathbf{e}) &= 0.192 \\ p(D = 0, E = 1 \mid G, \mathbf{e}) &= 0.112 \\ p(D = 0, E = 0 \mid G, \mathbf{e}) &= 0.248 \end{aligned}$$

を得る．また，Q の各変数ごとに周辺事後確率は，

$$\begin{aligned} p(D = 1 \mid G, \mathbf{e}) &= 0.64 \\ p(E = 1 \mid G, \mathbf{e}) &= 0.56 \end{aligned}$$

となる．

演習問題 6

”spam2”はテストデータ TD.csv は欠損を含んでいる．TAN を用いて spam2 の分類精度を測定せよ．ただし，TD.csv 内の -1 はデータの欠損を表している．

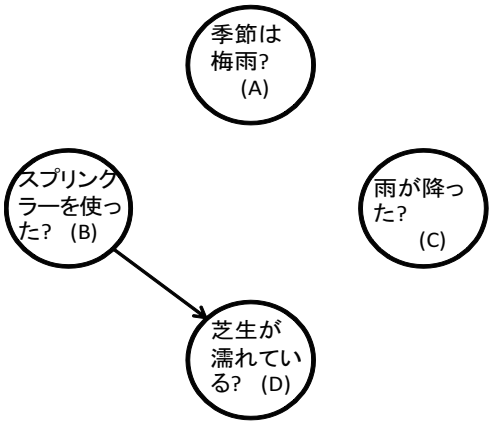
4.3.4 枝刈りによる高速化

ベイジアン･ネットワークにおいて，変数消去法によるエビデンス $\mathbf{e}$ を所与としたクエリ変数 Q の確率推論は，$(Q, \mathbf{e})$ の組み合わせにより枝刈り (pruning) を適用し，高速化することができる．すなわち，以下の定理が知られている．

定理 18 $(Q, \mathbf{e})$ を所与としたとき，ノード集合 $(Q \cup \mathbf{e})$ に含まれない葉ノード集合を削除できる．

定理 19 $(Q, \mathbf{e})$ を所与としたとき，ノード集合 $\mathbf{e}$ から張られた全てのエッジ集合を削除できる．

例 18 図 *3.12* において，$Q = \{D\}$, $\mathbf{e} : A =$ 真 *(1)*, $C =$ 偽 *(0)* が得られているとき，$(Q \cup \mathbf{e})$ に含まれない葉ノード集合はノード E のみであり，ノード E が削除される．さらに，ノード集合 $\mathbf{e}$ から張られた全てのエッジ集合は，ノード A, C から張られた全てのエッジを削除すればよいので，結果として図 *3.14* の縮約されたベイジアン・ネットワーク構造を得ることができる．変換されたベイジアン・ネットワークについて，$Q' = (Q \cup \mathbf{e})$ として $\mathbf{e}$ を新たなクエリ変数 Q' に組み込み，アルゴリズム *6* の周辺事前確率を求めるアルゴ



B	$\varphi^e(B)$
真	0.2
偽	0.8

C	$\varphi^e(C)$
真	0.8
偽	0.2

B　D	$\varphi^e(D \mid B)$
真　真	0.9
真　偽	0.1
偽　真	0.0
偽　偽	1.0

図 4.14: 図 3.12 についてクエリ $Q = \{D\}$, エビデンス $\mathbf{e} = \{A = 1, C = 0\}$ を得たときの枝刈りされたベイジアン・ネットワーク

リズムに $Q' \to Q$ として適用すればよい．結果として，$p(D, \mathbf{e} \mid G)$ は

$$
\begin{aligned}
p(D=1, \mathbf{e} \mid G) &= \sum_{B=\{0,1\}} \sum_{C=\{0,1\}} \sum_{A=\{0,1\}} p(A)p(B \mid A)p(C \mid A)p(D \mid B, C) \\
&= \left(\sum_{A=\{0,1\}} \varphi^e(A) \right) \left(\sum_{B=\{0,1\}} p(B \mid A)p(D \mid B, C) \right) \\
&\quad \times \left(\sum_{C=\{0,1\}} p(C \mid A) \right) \\
&= 0.6 \times 0.18 \times 0.2 = 0.0216 \\
p(D=0, \mathbf{e} \mid G) &= 0.6 \times 0.82 \times 0.2 = 0.0984
\end{aligned}
$$

同時確率を求めただけなので，以下のように事後確率を求めればよい．$p(\mathbf{e} \mid G) = 0.12$ なので，$p(D = 1 \mid G, \mathbf{e}) = \frac{0.0216}{0.12} = 0.18$, $p(D = 0 \mid G, \mathbf{e}) = \frac{0.0984}{0.12} = 0.82$ を得る．

第5章　ベイジアン・ネットワークの学習

これまで，ベイジアン・ネットワークにおける，エビデンスが与えられたときの未知変数の確率推論の問題を扱ってきた．では, ベイジアン・ネットワークはどのように構築すれば良いのであろうか？

この問題は**ベイジアン・ネットワーク学習**（Learning Bayesian networks）と呼ばれ，これまでにも膨大な量の研究が行われてきている．現在のこの分野では，ベイジアンネットワーク学習は構造の予測性の良さを示す学習スコアを最適にする構造を求める．

例えば，二変数の構造の候補は図 6.1 のように存在し，各構造のスコアを計算し，三つの中から最適なスコアを持つ構造を選べばよい．

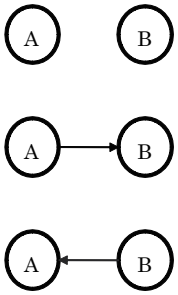


図 5.1: 二変数のベイジアン・ネットワーク構造候補

本章では，ベイジアン・ネットワーク学習のためのベイジアン・スコアおよびアルゴリズムを紹介する．

5.1　ベイジアン・ネットワークのパラメータ学習

3.3 で定義したベイジアン・ネットワークについて**統計的学習** (statistical learning) を可能にするために**母数化** (parameterization) する．今,N 個の離散変数集合 $x = \{x_1, x_2, \cdots, x_N\}$　について, 各変数が r_i 個の状態集合の中から一つの値をとるものとする. ここで, 変数 x_i が値 k をとるときに $x_i = k$ と書くことにし, $y = j$ を所与としたときの $x = k$ の条件付確率を $p(x = k \mid y = j)$ と書くことにする.

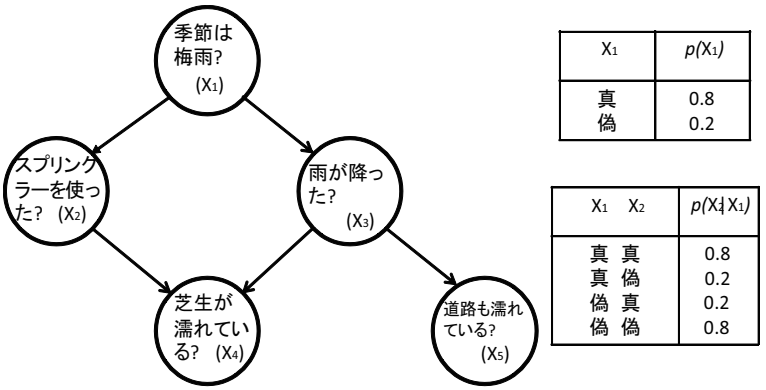
N 個の離散変数を持つベイジアン・ネットワークの同時確率分布は，条件付確率と確率の連鎖法則（チェーン・ルール）によって, 下のように示される.

$$p(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} p(x_i \mid x_1, \cdots, x_{i-1}) \tag{5.1}$$

ベイジアン・ネットワークでは，さらに確率構造 G を所与としているので，以下のように同時確率分布をモデル化できる (3.3 章).

$$p(x_1, x_2, \cdots, x_N \mid G) = \prod_{i=1}^{N} p(x_i \mid \Pi_i, G) \tag{5.2}$$

ただし，$\Pi_i \subseteq \{x_1, x_2, \cdots, x_{q_i}\}$ は変数 i の親ノード集合を示している.



X1	p(X1)
真	0.8
偽	0.2

X1 X2	p(X2|X1)
真 真	0.8
真 偽	0.2
偽 真	0.2
偽 偽	0.8

X1 X3	p(X3|X1)
真 真	0.8
真 偽	0.2
偽 真	0.2
偽 偽	0.8

X2 X3 X4	p(X4|X2,X3)
真 真 真	0.8
真 真 偽	0.2
真 偽 真	0.6
真 偽 偽	0.4
偽 真 真	0.6
偽 真 偽	0.4
偽 偽 真	0.2
偽 偽 偽	0.8

X3 X5	p(X5|X3)
真 真	0.8
真 偽	0.2
偽 真	0.2
偽 偽	0.8

図 5.2: ベイジアン・ネットワーク

例えば，図 6.2 の構造では，同時確率分布は以下のように計算できる.

$$p(x_1, \cdots, x_5 \mid G) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1)p(x_4 \mid x_2, x_3)p(x_5 \mid x_3) \tag{5.3}$$

今，θ_{ijk} を親変数集合が j 番目のパターンをとったとき $(\Pi_i = j)$ 変数 x_i が値 k をとる条件付き確率パラメータとし，条件付確率パラメータ集合 $\Theta = \{\theta_{ijk}\}, (i = 1, \cdots, N, j = 1, \cdots, q_i, k = 1, \cdots, r_i)$ とする．これらより，ベイジアン・ネットワークは，ネットワーク構造 G と条件付確率パラメータ集合 Θ によって (G, Θ) として表現できる.

このとき，データ $\mathbf{X}$ を所与としたときのパラメータ集合 Θ についての尤度は，以下のような多項分布に従うことがわかる．

$$\begin{aligned} p(\mathbf{X} \mid \Theta, G) &= \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{(\sum_{k=1}^{r_i} n_{ijk}\ )!}{\prod_{k=1}^{r_i} n_{ijk}!} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \qquad (5.4) \\ &\propto \prod_{i=1}^{N} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \end{aligned}$$

ここで，n_{ijk} は，変数 i の親ノード変数集合に j 番目のパターンをとり，変数 i に対して k 番目の値をとったデータ数を示している．$\frac{(\sum_{k=1}^{r_i} n_{ijk}\ )!}{\prod_{k=1}^{r_i} n_{ijk}!}$ は尤度を積分して 1.0 にする正規化項である．この尤度を最大化する母数 θ_{ijk} を求めることにより最尤推定値 $\widehat{\theta_{ijk}} = \frac{n_{ijk}}{n_{ij}}$ を求めることができる．しかし，よく知られているようにベイズ推定はより強力である．以下で紹介しよう．

5.1.1 事前分布

ベイズアプローチに従い，パラメータ Θ についての事前分布 $p(\Theta \mid G)$ を考えることにしよう．事前分布を考える場合，さまざまな考え方があるが，もっとも合理的であると考えられるのは，事前分布と事後分布の分布形が同一になるような事前分布，すなわち，自然共益事前分布（Cojecture prior）の導入であろう．上の尤度は**多項分布** (multinomial distribution) に従うので，その自然共益事前分布として以下の**ディレクレ分布** (Dirichlet distribution) が知られている．

$$p(\Theta \mid G) = \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1} \qquad (5.5)$$

ここで，Γ は $\Gamma(x+1) = x\Gamma(x)$ を満たすガンマ関数を示し，α_{ijk} は n_{ijk} に対応する事前の知識を表現する**疑似サンプル** (pseudo sample) としてのハイパーパラメータ (hyper parameters) を示す．

事前分布としてのディレクレ分布の意味について考えてみよう．ディレクレ分布は多数の母数を持つ多変量分布であるので，周辺分布である二値のみの変数ベータ分布 (Beta distribution) を用いて説明することにしよう．ベータ分布は 1 章ですでに説明したが, ここではもう一度復習しておこう．図 6.3 が母数値とベータ分布の形状の関係を示した図であり，横軸は条件付確率母数 θ_{ijk} に対応し，縦軸はその確率密度を示している．例えば，ハイパーパラメータ α_{ijk} がすべて 1 のとき，図にあるように θ_{ijk} の事前分布は一様分布 $\beta(1,1)$ になる．すなわち，データを得るまで母数 θ_{ijk} の推定値の分布は区間 $[0,1]$ の中で一様であり，すべて等しい．ハイパーパラメータ α_{ijk} がすべて $\frac{1}{2}$ のとき，図にあるように θ_{ijk} の事前分布は U 分布 $\beta(0.5, 0.5)$ になる．この分布は，無情報事前分布の一つであるジェフリーズの事前分布 (Jefrrs prior)

に一致することが知られ，マルコフ情報源で事前情報の無情報化を最大化することが証明されているし，情報理論ではミニマックス最適化が証明されている (Clark and Barron, 1994)．U 分布は，確率 0 と 1 のふたつの部分に二極化して密度が高くなり，0 か 1 かに近い値を条件付き確率がとることを仮定している．

また $\alpha_{ijk} = 10$ のように値を大きくすると，0.5 をモードとする凸分布に近付き，その値を大きくすればするほど推定値がデータの影響を受けにくくなる．

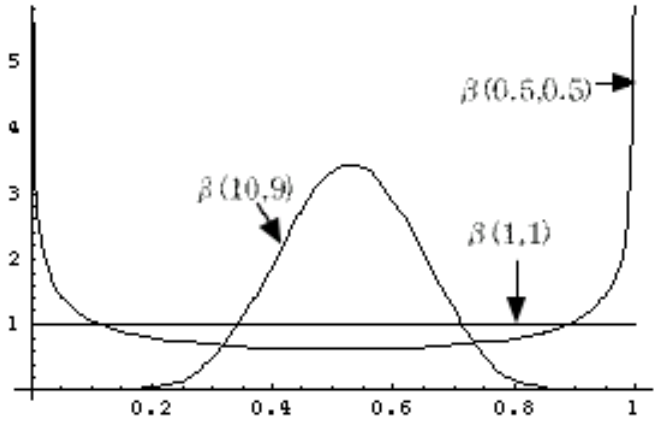


図 5.3: ディレクレ分布の周辺分布ベータ分布

5.1.2　母数推定

事後分布は，事前分布を尤度に掛け合わせることにより得ることができる．ベイジアン・ネットワークでは，先に求められた尤度とディレクレィ分布を掛け合わせると以下のような事後分布を得ることができる．

$$\begin{aligned} p(\mathbf{X}, \Theta \mid G) &= \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma\left(\sum_{k=1}^{r_i} \alpha_{ijk}\right)}{\prod_{k=1}^{r_i} \Gamma\left(\alpha_{ijk}\right)} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}+n_{ijk}-1} \qquad (5.6) \\ &\propto \prod_{i=1}^{N} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}+n_{ijk}-1} \end{aligned}$$

この事後分布を最大にする MAP(Maximum A Posteriori) 推定値は，以下の対数事後分布を最大にする母数の推定値を求めればよい．

$$\log p(\mathbf{X}, \Theta \mid G) = \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} (\alpha_{ijk} + n_{ijk} - 1) \log \theta_{ijk} + const \qquad (5.7)$$

$\sum_{k=1}^{r_i} \theta_{ijk} = 1$ より，Lagrange 乗数法を用いる．等式制約 $\sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0$ を含んだラグランジェ関数 L は，

$$\begin{aligned} L &= \log p(\mathbf{X}, \Theta \mid G) + \lambda \left(\sum_{k=1}^{r_i} \theta_{ijk} - 1 \right) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} (\alpha_{ijk} + n_{ijk} - 1) \log \theta_{ijk} + \lambda \left(\sum_{k=1}^{r_i} \theta_{ijk} - 1 \right) \end{aligned} \tag{5.8}$$

L を θ_{ijk} について偏微分し，0 となる θ_{ijk} を求めればよい．すなわち，

$$\frac{\partial L}{\partial \theta_{ijk}} = \frac{\alpha_{ijk} + n_{ijk} - 1}{\theta_{ijk}} + \lambda = 0 \tag{5.9}$$

を解けばよい．(6.9) 式から

$$\widehat{\theta_{ijk}} = \frac{\alpha_{ijk} + n_{ijk} - 1}{-\lambda} \tag{5.10}$$

また，(6.9) 式を変形して

$$\sum_{k=1}^{r_i} (\alpha_{ijk} + n_{ijk} - 1) = -\sum_{k=1}^{r_i} (\theta_{ijk} \lambda) \tag{5.11}$$

(6.10) より，

$$\lambda = -(\alpha_{ij} + n_{ij} - r_i) \tag{5.12}$$

ここで，$\alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk}$, $n_{ij} = \sum_{k=1}^{r_i} n_{ijk}$．結果として，以下の MAP 推定値を得る．

$$\widehat{\theta_{ijk}} = \frac{\alpha_{ijk} + n_{ijk} - 1}{\alpha_{ij} + n_{ij} - r_i} \text{最尤推定値 } (Maxim \tag{5.13}$$

MAP 推定値では，全てのハイパーパラメータを $\alpha_{ijk} = 1$ として一様分布に設定すると um Likelihood estimator) に一致する．

しかし，ベイズ統計学では，MAP 推定値よりも事後分布の期待値である EAP(Expected A Posteriori) 推定値のほうが頑健で予測効率がよいことが知られている．ディリクレ分布，式 (6.6) の期待値は

$$\widehat{\theta_{ijk}} = \frac{\alpha_{ijk} + n_{ijk}}{\alpha_{ij} + n_{ij}} \tag{5.14}$$

となる．ベイジアン・ネットワーク学習では，EAP 推定値が最も一般的に用いられる．

5.1.3 数値例

表 6.1 のデータは，図 6.2 の因果モデルから発生させた 20 個の事例サンプルである．このような事例データから，元の因果モデルを推定することが，本章の目的であるベイジアン・ネットワークの因果モデルの学習である．

表 5.1: データ例

	ノード番号				
	1	2	3	4	5
1	1	0	1	1	1
2	0	0	0	1	0
3	1	1	1	1	1
4	1	1	1	1	0
5	1	1	1	1	0
6	1	1	0	0	0
7	1	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	1	1	1	1
11	0	0	0	0	0
12	1	1	1	1	1
13	0	0	1	1	1
14	1	1	1	1	1
15	1	0	1	1	1
16	1	1	1	0	0
17	1	1	0	1	0
18	1	1	1	0	0
19	1	1	0	1	0
20	1	1	1	0	1
平均	0.70	0.60	0.60	0.60	0.40

表 6.3 に，ハイパーパラメーター α_{ijk} を変化させたときの各ハイパーパラメータの値に対応する推定値と真の値との平均自乗誤差を示した．

最尤推定値，事前分布に一様分布を仮定した $\alpha_{ijk} = 1$ ジェフリーズ事前分布 $\alpha_{ijk} = 1/2$ を用いた数値例を示している．それぞれ異なる推定値を得ていることがわかる．ただし，データ数が少ないので，ここでの予測成績結果は一般的なものではない．

5.2　周辺尤度による構造学習

ベイジアンネットワーク構造 G をデータから推測する問題はパラメータ学習より複雑である．一般には，構造 G の周辺尤度 (Marginal likelihood) を最大化する構造を見つければよい．ただし，ベイズ分野では周辺尤度最大化による推定が必ずしも望ましい目標とは考えられないこともある．特に工学分

表 5.2: データ例

	真の値	最尤推定値	$\alpha_{ijk}=1$	$\alpha=1/2$
$p(x_2=1 \mid x_1=1)$	0.8	0.8	0.76	0.78
$p(x_2=1 \mid x_1=0)$	0.2	0	0.14	0.08
$p(x_3=1 \mid x_1=1)$	0.8	0.8	0.76	0.78
$p(x_3=1 \mid x_1=0)$	0.2	0	0.14	0.08
$p(x_4=1 \mid x_2=1,x_3=1)$	0.8	0.66	0.64	0.65
$p(x_4=1 \mid x_2=1,x_3=0)$	0.6	0.5	0.5	0.5
$p(x_4=1 \mid x_2=0,x_3=1)$	0.6	0.5	0.5	0.5
$p(x_4=1 \mid x_2=0,x_3=0)$	0.2	0.4	0.43	0.41
$p(x_5=1 \mid x_3=1)$	0.8	0.67	0.64	0.42
$p(x_5=1 \mid x_3=0)$	0.2	0.33	0.35	0.34
真の値との平均自乗誤差		0.019	0.015	0.028

野の場合，因果構造を推定するためなのか，エビデンスを得た後の確率推論のためなのか，分類のためなのかなどで望ましい構造は異なってくるのである (Chickering and Heckerman 2000)．しかし，少なくとも真の確率的因果構造を推定するという目的に関しては周辺尤度最大化が有用である．ベイジアンネットワークの予測分布は，事後分布よりパラメータ推定値を周辺化した周辺尤度 (ML:Marginal Likelihood) として以下のように得られる．

$$\begin{aligned} p(\mathbf{X} \mid G) &= \int_{\Theta} p(\mathbf{X} \mid \Theta, G) p(\Theta) d\Theta \\ &= \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+n_{ij})} \prod_{k=1}^{r_i} \frac{(\alpha_{ijk}+n_{ijk})}{\Gamma(\alpha_{ijk})} \end{aligned} \tag{5.15}$$

周辺尤度には, ユーザーが事前に決定しなければならない事前分布のハイパーパラメータ α_{ijk} が残っていることがわかる．ユーザーが事前に構造に関する知識を持っているのであれば, その構造から生起される擬似データを α_{ijk} として与えれば良い．しかし, 一般には事前に知識はなく, なるべく偏見なく構造を推定したい．そのためには 1 章で述べた無情報事前分布を用いなければならない．1 章で述べたように, ベイズ統計学における無情報事前分布は多種あり, どれがよいかも場合によって異なる．そのために, ベイジアン・ネットワークでもハイパーパラメータの決定については多く議論されている．

例えば,Cooper and Herskovits(1992) は, 無情報事前分布として一様分布 (For $\forall i, \forall j, \forall k, \alpha = 1.0$) を考え，以下の周辺尤度スコアを提案している．

$$p(\mathbf{X} \mid G) \propto \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{(r_i-1)!}{(n_{ij}+r_i-1)!} \prod_{k=0}^{r_i-1} n_{ijk}! \tag{5.16}$$

このスコアは構造発見アルゴリズム ”K2” (Cooper and Herskovits, 1992) として実装されている．

5.3　最小記述長 (MDL) による学習

最小記述長 (minimum description length:MDL) は Rissanen(1978) によって提案されたフレームワークで，モデルとデータの同時記述長を最小化するというものである．最初のベイジアンネットワークの MDL 符号化は, Lam and Bacchus(1994) によって提案されている．しかし，彼らの符号化はデータ数の関数でなく，以下のクラフトの不等式を満たさず記述長ではない．

定義 33 関数 $L(g, \mathbf{X})$ は

$$\sum_{\boldsymbol{X}\in\chi^n} 2^{-L(g,\mathbf{X})} \leq 1.$$

(Kraft's inequality) for any G を満たすとき，記述長 *(description length)* である．

Suzuki(1998) は，クラフトの不等式を満たす記述長として，ハイパーパラメータが $\alpha_{ijk} = \frac{1}{2}, (i = 1, \cdots, N, j = 1, \cdots, q_i, k = 1, \cdots, r_i)$ の条件を満たすとき，以下の MDL 符号を提案している．

$$I(G, \mathbf{X}) = \ln p(G) + \sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\left[n_{ijk}\ln\frac{n_{ijk}}{n_{ij}}\right] - \frac{\sum_{i=1}^{N} q_i(r_i-1)}{2}\ln n \tag{5.17}$$

ここで，$n = \sum_{j=1}^{q_i} n_{ij}$ である．このスコアはベイジアン情報量基準 **BIC** (Bayesian Information Criterion, Schwarz,G.E. (1978) に一致するので強一致性を持ち，漸近的に真の構造を推定できる保証がある．Suzuki(1998) は，すべてのハイパーパラメータが $\alpha_{ijk} = \frac{1}{2}$ を満たすときにベイジアンネットワーク構造の対数周辺尤度 (6.15) は (6.16) の MDL 符号に収束することを証明している．前述のように，情報理論では $\alpha = \frac{1}{2}$ のときのディレクレ分布は，ミニマックス基準で最適値をとることが知られていたので，Suzuki(1998) の展開は情報理論分野として一貫している．しかし，Bouckaert, R. (1994a, 1994b) は, すべてのハイパーパラメータが $\alpha_{ijk} = 1.0$(すなわち，事前分布が一様分布) を満たすときにベイジアンネットワーク構造の対数周辺尤度 (6.15) は (6.16) の MDL 符号に収束することをすでに証明している．この論争は興味深いが，Ueno and Kubo(2008) はどちらの導出も正しく，α_{ijk} をデータ数より十分小さく設定すると, どのような α_{ijk} についても対数周辺尤度 (6.15) は (6.16) の MDL 符号に収束することを証明している．

その後,Yang and Chang (2002) は, シミュレーションにより (6.16) の MDL 符号とハイパーパラメータを変化させた周辺尤度スコア (6.15) の学習成績を比較した結果，MDL 符号は有意に劣った学習性能を持っていると報告している．この理由は，対数尤度 $\sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=0}^{r_i-1}\left[n_{ijk}\ln\frac{n_{ijk}}{n_{ij}}\right]$ が事前知識を用いていないため，スパースなデータの場合オーバーフィッティングを引き起こしてしまうことにある．

Ueno and Kubo(2008) は，先の MDL 符号が特定の $\alpha_{ijk}(1/2$ や $1)$ に固定して対数周辺尤度を漸近展開していることに着目し，α_{ijk} を固定せずにクラフトの不等式を満たすように対数周辺尤度を漸近展開し，事前知識を反映した以下のようなベイジアンネットワーク MDL 符号を導出している．

$$\begin{aligned} I(G,\mathbf{X}) &= \sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=0}^{r_i-1}\left[(n_{ijk}+n'_{ijk})\ln\frac{n_{ijk}+n'_{ijk}}{n_{ij}+n'_{ij}}\right] \\ &\quad -\frac{\sum_{i=1}^{N}q_i(r_i-1)}{2}\ln\frac{n'+n}{2\pi} \end{aligned} \tag{5.18}$$

この MDL 符号は学習効率を改善している．

上の MDL 符号は BIC と同様に対数周辺尤度の近似に他ならない．Rissanen も指摘しているが，MDL を導出した文献 Rissanen(1978) は，MDL と BIC が同値であるという誤解を与えてしまっている．最も重要な MDL 原理の要素は，モデルクラス (model class) に関するデータの最小記述長，**Stochastic Complexity** である (Rissanen, 1987)．モデルクラス $\mathcal{M}$ とは，d 次元パラメータ空間 $\Theta\in\mathcal{R}^d$ の要素によって以下のように定義される．

定義 34

$$\mathcal{M}=\{P(\cdot\mid\theta):\theta\in\Theta\} \tag{5.19}$$

このとき，**NML**(Normalized Maximum Likelihood)(Shtarkov, 1987) は以下のように定義される．

定義 35 $\mathbf{X}$ を n 個のデータ系列とすると *NML* は以下に定義される．

$$p(\mathbf{X}\mid\mathcal{M})=\frac{p(\mathbf{X}\mid\widehat{\theta},\mathcal{M})}{\mathcal{R}_{\mathcal{M}}}, \tag{5.20}$$

ここで

$$\mathcal{R}_{\mathcal{M}}=\sum_{x^N}p(\mathbf{X}\mid\widehat{\theta},\mathcal{M})$$

$\mathcal{R}$ はリグレット (regret) と呼ばれる．この NML を最大とするモデルがデータ系列 $\mathbf{X}$ の最小符号語長に一致し，ミニマックス最適化を保証する (Roos and Rissanen, 2008)．しかし，長さ n のデータの全パターン $\sum_{x^N}$ の計算が必要で計算量が指数オーダーであることが問題であった．ベイジアン・ネットワークでは，r 個の値をとる多項分布に従うので，ノード i についての $\mathcal{R}$ は

$$\mathcal{R}=\prod_{j=1}^{q_i}\sum_{D'_{ij}}\prod_{k=1}^{r_i}\left(\frac{n_{ijk}}{n_{ij}}\right)^{n_{ijk}}, \tag{5.21}$$

で示され (Silander, Roos, Myllymaki,2009)，ここで D'_{ij} はノード i が親ノードパターン j を持つときデータ集合を示している．

5.4　尤度等価と BDe(u) スコア

Heckerman,Geiger and Chickering(1995) は, 二つの構造が等価であるならそれらのパラメータ同時確率密度は同一でなければならないという「尤度等価」(likelihood equivalence) 原理をベイジアン・ネットワーク学習に導入した．理論的には，尤度等価は次のように定義される．

定義 36 *(尤度等価)* $p(G_1 \mid \xi) > 0$ かつ $p(G_2 \mid \xi) > 0$ となるような構造 G_1 と G_2 を所与として，G_1 と G_2 が等価であるならば，$p(\Theta_{G_1} \mid G_1, \xi) = p(\Theta_{G_2} \mid G_2, \xi)$ である．これを満たすように構造学習できるスコアを「尤度等価 *(likelihood equivalence)* を満たす」と呼ぶ．

Heckerman,Geiger and Chickering(1995) は，前述の K2,　すなわち一様事前分布 $\alpha_{ijk} = 1.0$ とした周辺尤度スコア (6.15) やミニマックス基準最適な $\alpha_{ijk} = \frac{1}{2}$ が尤度等価を満たさないことを指摘し，尤度等価の条件としてハイパーパラメータの和が式 (6.15) で一定となる以下の条件を導出し，これを満たすスコアを BDe(Bayesian Dirichlet equivalence) と呼んでいる．

$$\alpha_{ijk} = \alpha p(x_i = k, \Pi_i = j \mid G^h) \tag{5.22}$$

ここで α は,「equivalent sample size(ESS)」と呼ばれる事前知識の重みを示す疑似データである．G^h は，ユーザーが事前に考えているネットワーク構造の仮説であり，その構造を所与として ESS を α_{ijk} に分配して事前知識を MAP 推定値に反映させる．

Buntine(1991) はすでに $\alpha_{ijk} = \alpha/(r_i q_i)$ とし，ESS をパラメータ数で除したスコアを提案している．このスコアは BDe の特別なケースとして捉えることができ「BDeu」と呼ばれる．Heckerman,Geiger and Chickering(1995) も指摘しているが，ユーザーが事前にネットワーク構造を書くのは難しいし，誤っている可能性も高いので，現実的には BDeu を用いることが望ましい．

実際，現在の最先端研究で用いられるベイジアンネットワークの学習スコアは BDeu である．

5.5　ディレクレ・スコアのハイパーパラメータ

近年，研究が進み，前節で紹介したディレクレ・スコアの性質がより詳細に明らかにされてきている（Steck and Jaakkola (2002), Silander, Kontkanen, and Myllymaki (2007), Ueno(2010), Ueno(2011)）．特に，近年の研究はディレクレ・スコアのハイパーパラメータの設定が学習効率に非常に重要な役割を担うことが示されてきた．

5.5.1 周辺尤度スコア

周辺尤度スコアは K2 や BDe(u) の基本となるスコアであり，基本的性質を理解することは非常に重要である．ここでは，Ueno(2010) によって導かれた以下の重要な定理を紹介する．

定理 20 *(事前知識によって決定される学習: Ueno(2010))* $\alpha + n$ が十分大きいとき，対数周辺尤度スコア *(式 (6.15))* は以下に近似される．

$$\log p(\boldsymbol{X} \mid \alpha, G) = \sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}(\alpha_{ijk}+n_{ijk})\log\frac{\alpha_{ijk}+n_{ijk}}{\alpha_{ij}+n_{ij}} \quad (5.23)$$
$$-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\frac{r_i-1}{r_i}\log\left(1+\frac{n_{ijk}}{\alpha_{ijk}}\right)+O(1).$$

式 (6.20) から，対数周辺尤度スコアが通常のモデル選択基準のようにネットワーク構造のデータへのあてはまりを反映する対数尤度の項 $\sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}(\alpha_{ijk}+n_{ijk})\log\frac{\alpha_{ijk}+n_{ijk}}{\alpha_{ijk}+n_{ij}}$ とネットワーク構造の複雑さを反映するペナルティ項 $\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\frac{r_i-1}{r_i}\log\left(1+\frac{n_{ijk}}{\alpha_{ijk}}\right)$ のトレードオフになっていることがわかる．式 (6.22) より，ユーザーが仮定している事前の構造 G^h とそれへの信念の強さ α が決定すれば α_{ijk} が決定され，α_{ijk} は事前分布を意味する．従って，$\sum_{i=1}^{N}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i}\frac{r_i-1}{r_i}$ はパラメータ数であり，$\log\left(1+\frac{n_{ijk}}{\alpha_{ijk}}\right)$ はデータ（経験分布）と事前分布の差異を示しており，このペナルティ項はユーザーの事前知識とデータの差異に重みづけられたパラメータ数のペナルティ項である．すべてのデータで α_{ijk} と n_{ijk} が一致すれば，パラメータ数へのペナルティは最小になる．ユーザーの知識に基づく構造がデータと離れているとパラメータのペナルティ項の重みはより大きくなり，逆にユーザーの事前知識が真の構造に近くなるとパラメータペナルティ項の重みはより小さくなるという仕組みである．すなわち，ユーザーの事前知識とその信念の強さが学習を決定しているのである．

ユーザーの事前知識がベジアンネットワーク学習を決定しているとすると，従来の AIC や BIC(MDL) などのモデル選択基準も対数周辺尤度スコアによって表現できると考えられる．このアイデアより以下の定理が導かれる．

定理 21 *(AIC の事前知識表現: Ueno 2010) For* $\forall i, \forall j, \forall k$, $\alpha_{ijk} = \frac{1}{3}n_{ijk}$ *のとき, 対数周辺尤度スコアは AIC により* $\mathcal{O}(1)$ *で近似される．*

この定理は，AIC がクロスバリデーションの近似として知られているが，実はそうはなっていないということを示している．例えば，クロスバリデーションの考え方に基づき，式 (6.23) において事前に得られたデータより計算された α_{ijk} を事前知識として代入し，新たに得られたデータから n_{ijk} として式 (6.23) を計算すると，仮説となる構造が正しければ α_{ijk} と n_{ijk} の分布の差異が小さくなり，パラメータ数へのペナルティ項も小さくなるのでスコアは最

大になりやすい．一方，仮説となる構造が真の構造と異なれば，α_{ijk} と n_{ijk} の分布の差異が大きくなり，パラメータ数へのペナルティ項も大きくなるので結果としてスコアは小さくなる．このようにしてクロスバリデーションに基づき，経験的に α_{ijk} を決定することの有用性は認められるが，定理 35 より，実際の AIC では α_{ijk} も n_{ijk} を学習させた同一のトレーニングデータで学習させていることになる．このことは以下のように過学習（オーバーフィッティング：ベイジアンネットワークにおいて真の構造に対して不要なエッジを多くつけてしまうこと）を引き起こす重要な原因となる．α_{ijk} が n_{ijk} に一致した場合，式 (6.23) の対数事後分布は n_{ijk} を基にした対数尤度に一致し，α_{ijk} のデータは全く尤度に影響を与えていないことになる．しかし，パラメータ数のペナルティ項は α_{ijk} と n_{ijk} が完全一致するために小さくなってしまい，対数尤度はエッジの追加に対して単調に増加してしまうので AIC は絶えず過学習の傾向が強くなってしまうのである．

また，BIC との関係についても以下の定理が導かれている．

定理 22 *(BIC の事前知識表現:Ueno (2010)) For* $\forall i, \forall j, \forall k$, $\alpha_{ijk} = 1.0$ （事前分布が一様分布*, K2(Cooper and Herskovits, 1992)* に一致）のとき*,* 対数周辺尤度スコアは *BIC* により $\mathcal{O}(1)$ で近似される．

すなわち，$\alpha_{ijk} = 1.0$ のときの周辺尤度スコアは BIC とほぼ同様に振る舞うことになる．(Bouckaert, 1994a,1994b) は $\alpha_{ijk} = 1.0$ のときの対数周辺尤度が MDL(6.17) に一致することを示しており，この結果と合致する．

また，Ueno(2010) は，真の構造における条件付き確率分布が一様分布に近づくと，式 (6.23) における α_{ijk} が一定の場合，ペナルティ項の重みがデータ数に対して一定になるので BIC は AIC の振る舞いに類似してくる性質についても言及している．

5.5.2　BDeu スコア

先に述べたように BDeu は最も有力なベイジアンネットワークの学習スコアである．近年，BDeu の研究が深く進められ，ベイジアンネットワーク学習において ESS の値が非常に重要な役割を果たすことが報告されている．

図 6.3 よりわかるように BDeu では事前分布としてのディレクレ分布で ESS, α の値を大きくすると $1/r_i$ をモードとした凸分布の尖度が大きくなっていく．すなわち事前分布は条件付き確率母数の値をエントロピーを最大にする $1/r_i$ へ近づけようとすることになる．直観的には，α を大きくすると構造学習ではエッジを削除しようとする方向に働くと考えられる．

Steck and Jaakkola(2002) は，シミュレーションを行い，データ数が大きいときでさえ，ESS を 0 に近づけていくと学習されたベイジアンネットワークのエッジ数が減っていき，ESS を大きくするとエッジ数が増加することを

発見している．ESS が 0 に近づくと条件付き確率パラメータのベイズ推定値は最尤推定値に近づき，ESS が大きくなると推定値の尖度が緩慢になるので，ESS が大きくなるとベイジアンネットワーク学習におけるエッジ追加のペナルティが大きくなると信じられてきた．しかし，Steck(2002) のシミュレーションで示された現象は全く逆であり，驚くべき事実を発見したことになる．

Silander,Kontakanen,and Myllymaki(2007) は，このメカニズムをより詳細に調べるために多くの実データを用いて検証している．実データでも Steck(2002) 同様の現象が確認されている．さらに彼らは BDeu の学習効率が ESS の値設定に非常に敏感であることを指摘し，その解決手法の一つとして ESS の値を 1 から 100 まで 1.0 ずつ変化させながら BDeu を最大化させる ESS を見つける経験ベイズ手法を提案している．

Steck(2002) は，一つのエッジの付与か否かの予測分布の比についてベイズファクター (Bayes factor) を漸近展開し，AIC(Akaike Information Criterion) Akaike(1974) を最小化にする ESS を解析的に求める経験ベイズ手法を提案している．Silander,Kontakanen,and Myllymaki(2007) の経験ベイズ手法とほぼ同等の精度を保ったまま高速に最適な ESS 値が求められることが示されている．

これらの研究では，BDeu を最大にする ESS を求める手法であったが，Ueno(2008) はそれらの手法が実際に予測を最大にする ESS 値とずれることを示している．すなわち，BDeu を最大にする ESS は予測効率を最大にする保証はないことになる．そこで，Ueno(2008) はクロスバリデーションと山登り法を組み合わせ予測を最適化する ESS 値を求めることを提案し，実験により有効性を示している．

系 1 *(log-BDeu の ESS によるトレードオフ:Ueno2010)* $\alpha + n$ が十分大きいとき，*log-BDeu* は以下に近似できる．

$$\log p(\boldsymbol{X} \mid G) = \alpha \sum_{i=1}^{N} \log r_i + \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=0}^{r_i-1} (\frac{\alpha}{r_i q_i} + n_{ijk}) \log \frac{\frac{\alpha}{r_i q_i} + n_{ijk}}{\frac{\alpha}{q_i} + n_{ij}}$$
$$-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=0}^{r_i-1} \frac{r_i - 1}{r_i} \log \Big(1 + \frac{r_i q_i n_{ijk}}{\alpha}\Big) \qquad (5.24)$$

ここで，$\alpha \sum_{i=1}^{N} \log r_i$ はエッジの追加に対して一定の値をとる．従って，log-BDeu は，(1) 対数尤度 $\sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=0}^{r_i-1} (\frac{\alpha}{r_i q_i} + n_{ijk}) \log \frac{\frac{\alpha}{r_i q_i} + n_{ijk}}{\frac{\alpha}{q_i} + n_{ij}}$ と (2) パラメータ数に対するペナルティ項 $\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=0}^{r_i-1} \frac{r_i - 1}{r_i} \log \big(1 + \frac{r_i q_i n_{ijk}}{\alpha}\big)$ に分解できる．これはモデル選択基準形式としてよく知られた (1) データへの当てはまりを反映する項と (2) 余計なエッジを追加しないように働くペナルティ項として解釈できる．

また，式 (6.24) の右辺において ESS が増加すれば，第一項では対数事後分布を一様分布に近づけエッジをつけないように働き，第二項ではペナルティ

を減少させエッジを増加させようとする働きをすることがわかる．すなわち，第一項と第二項で ESS によるエッジの増減に対するトレードオフが存在することになり，これが先行研究で発見されてきた ESS に対して BDeu が非常に敏感に反応する原因となっていることがわかる．

また，式 (6.24) より，トレードオフが解消される漸近的な状況では以下の性質が導かれる．

定理 23 *(ESS* 増大による完全グラフ生成*:Ueno (2010)) ESS* を大きくしていくと，学習されたベイジアンネットワーク構造のエッジ数は単調増加し，完全グラフ *(Complete Graph)* に近づいていく．

定理 24 *(ESS* 減少による空グラフ生成*:Ueno (2010)) ESS* を小さくしていくと，学習されたベイジアンネットワーク構造のエッジ数は単調減少し，空グラフ *(Empty Graph)* に漸近的に近づいていく．

以上示したように, **系 2** により,Steck(2002) により発見された BDeu 学習の原因不明の現象のメカニズムが明らかになった．

さらに**系 2** により，BDeu において最適な ESS 設定についても以下のような知見を得ることができる．

- **真の条件付き確率分布が非一様のとき**：最適な ESS 値は比較的小さい値を与えなければならない．なぜならば，式 (6.24) 右辺において，対数事後分布はエッジを追加しようとするので過学習を起こしやすい．第二項で過学習を抑えるために ESS 値は小さい値に設定しなければならない．

- **真の条件付き確率分布がほぼ一様のとき**：最適な ESS 値は比較的大きな値を与えなければならない．なぜならば，真の条件付き確率分布が一様なので式 (6.24) 右辺において，対数事後分布は真の因果を発見することが難しい．第二項で過小学習 (underfitting) を避けるために ESS 値は大きい値に設定しなければならない．

- **スパース（過疎）なデータのとき**：n_{ijk} が欠測データもしくは過疎である場合，なるべく ESS の影響を抑え，データの学習への影響を最大にしなければならない．式 (14) より ESS=1.0 となるとき，データの影響を最大化できる．Castillo., Hadi, and Solares (1997) は周辺尤度の分散が n_{ijk} が欠測のとき，ESS の二乗に反比例することを示している．すなわち，この意味でも ESS=1.0 がデータの影響を最大にできることがわかる．

また，α の値を小さくするとペナルティ効果が加速することが知られている．

定理 25 *(ESS* 減少による空グラフ生成*:Ueno (2011))*
$\alpha < r_i q_i, (i = 1, \cdots, N)$ *,* n が十分大きいとき*,* 対数周辺尤度は以下に収束

する．

$$\log p(\mathbf{X} \mid \mathbf{g}, \alpha) = \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=0}^{r_i - 1} (\frac{\alpha}{r_i q_i} + n_{ijk}) \log \frac{\frac{\alpha}{r_i q_i} + n_{ijk}}{\frac{\alpha}{q_i} + n_{ij}}$$
$$- \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{q_i} \left(\frac{(r_i - 1)}{r_i} \sum_{k=1}^{r_i} \log(\frac{(r_i q_i)^2 n_{ijk}}{2\pi\alpha^2} \right) + \mathcal{O}(1) \quad (5.25)$$

α の値が小さくなると，ペナルティ項の分母が α^2 となり，ESS を小さくすることの効果を増幅させ，ESS の設定に対してより敏感になる．

このように BDe(u) では ESS の値によって学習パフォーマンスが非常に敏感に変化してしまう．そこで ESS をどのように設定するかが, 重要となる．Ueno (2010) は以下の定理を導いている．

定理 26 *(最適な ESS の決定:Ueno (2010)) BDe(u) は,ESS(α)*= 1.0 *のとき,* 事後分布の分散を最大化する*.*

すなわち, できる限りデータの影響を最大にするようにするためには,ESS= 1.0 が最適であることがわかる．特に BDeu を用いるときには重要な注意点となる．

5.6 無情報事前分布による学習スコア

前節で考察したように, 現在, 最もよく用いられる BDeu は無情報事前分布を用いず, 条件付き確率パラメータの事前分布に一様分布を仮定してしまっている．そのことが BDeu の学習の頑健さを損なう原因ともなっている．ディレクレ・スコアの尤度等価を満たすように無情報事前分布を持つ学習スコアをつくることは難しい．Ueno(2011) は, 対数 BDeu の事前分布の偏りを近似的に削除して, 新しい学習スコア **NIP-BIC**(Non Informative Prior Bayesian Information Criterion) を提案している．

定義 37 *NIP-BIC (Ueno 2011)*

$$NIP - BIC = \sum_{i=1}^{N} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} (\alpha_{ijk} + n_{ijk}) \log \frac{\alpha_{ijk} + n_{ijk}}{\alpha_{ij} + n_{ij}} \quad (5.26)$$
$$- \frac{1}{2} q_i r_i \log(1 + n),$$

また, 厳密な無情報事前分布によるディレクレ･スコア **NIP-BDe**(Non-informative prior Bayesian Dirichlet equivalence) が BDe の事前分布における全ての仮説構造について周辺化することにより, 以下のように提案されている．

定義 38 完全無情報事前分布スコア *NIP-BDe (Ueno 2012)*

$$p(\boldsymbol{X} \mid g, \alpha) = \sum_{g^h \in G} p(g^h) p(\boldsymbol{X} \mid \alpha, g, g^h) \quad (5.27)$$

$$= \sum_{g^h \in G} p(g^h) \left(\prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij}^{g^h})}{\Gamma(\alpha_{ij}^{g^h} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk}^{g^h} + n_{ijk})}{\Gamma(\alpha_{ijk}^{g^h})} \right),$$

ここで $\sum_{g^h \in G}$ はすべての可能な構造候補に対する和, $p(g^h)$ は構造候補の事前分布でここでは一様分布を仮定している.

これらは BDeu に対し, 学習を大きく改善できることが知られている．ただし，厳密な無情報事前分布スコア NIP-BDe は計算量が非常に大きく, NIP-BDe は漸近的に NIP-BIC に近似できることが証明されており (Ueno2012), 実用的には NIP-BIC が推薦されるかもしれない．しかし, 研究レベルでは厳密な無情報事前分布スコアの計算は非常に注目されており, 高速なアルゴリズム開発はベイジアン・ネットワークの重要な研究課題である.

5.7　構造の探索アルゴリズム

前節までは，ベイジアン・ネットワーク構造の学習スコアについて紹介したが，残る問題は如何にして全ての考えられる構造から最適な構造を選択するということである.

図 6.4 は，3 変数の可能な無向グラフの数を示している．3 変数でさらにエッジの方向を考えなくても 8 つの構造が存在することがわかる．[?] では，変数数 N が既知の場合のベイジアン・ネットワーク（有向グラフ）の構造数の計算方法を以下のように提案している.

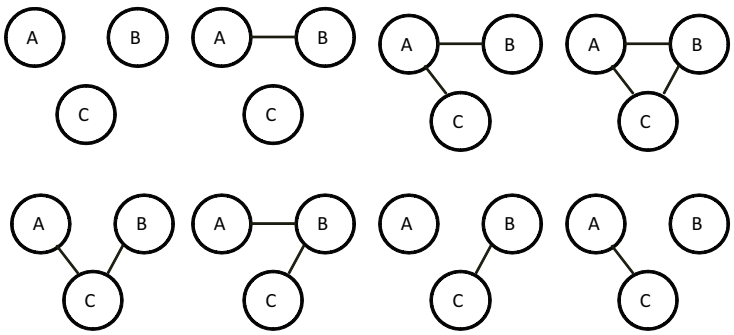


図 5.4: 3 変数の場合の構造候補数

$$f(N) = \sum_{i=1}^{N} (-1)^{i+1} \left[\begin{array}{c} n \\ i \end{array} \right] 2^{i(N-i)} f(N-i) \tag{5.28}$$

例えば，$N = 2$ のとき構造数は 3，$N = 3$ のとき構造数は 25，$N = 5$ で $29,000$，$N = 10$ でおよそ 4.2×10^{18} となる．この探索問題は NP 完全問題

(Chickering, 1996) であり，なんらかの工夫により探索計算量を減じなければならない．

5.8 厳密解探索アルゴリズム

ベイジアン・ネットワークの構造探索問題は NP 完全であることが知られている (Chickering, 1996)．しかし，近年，BDeu などのスコアを最大化するための**厳密解** (Exact Slution) を現実的な時間で得る手法が提案されてきている (Koivisto and Sood, 2004), (Silander and Myllymaki, 2006) ．具体的には，K2 で採用されていた**貪欲法** (Greedy Search) におけるノード順序を全ての組み合わせで計算するというアイデアである．すべてのノード順序についてのローカルスコアの和が必要になるが，Silander and Myllymaki, 2006) では, 動的計画法を用いて導出しているのが特徴であり，最初に BDeu スコアをローカルスコアに分解し，それを用いて最適なノード順序を探索する手法を提案している．Perrier, Imoto and Miyano (2008) は，新しい構造制約として必ず真の構造を含んでいる無向グラフ **super-structure** を提案している．実際には super-structure は条件付き独立検定法を用いてその有意確率を 95 % 程度まで上げ，ほぼ確実に真の構造を含む無向グラフをスケルトンとして与え，それを制約として BDeu などのスコア法での厳密解を求める手法を提案している．これにより，大きく計算量を減らすことができる場合があることを示している．また，Ordyniak and Szeider(2010) は，super-structure の適用についての計算量についてその下限値を分析し，super-structure が適応した場合，かなり計算量を減じることができることを示している．

本節では，Silander and Myllymaki(2006) によって提案された**動的計画法** (Dynamic Programming, DP) による厳密解探索アルゴリズムを紹介する．

Silander and Myllymaki(2006) のアルゴリズムでは，最初に，あり得る全てのサブネットワーク (x_i, Π_i) のスコア (ローカルスコア) を計算する．サブネットワークのパターンは，N 個の離散変数を持つベイジアン・ネットワークでは，$N2^{N-1}$ 個考えられる．

ローカルスコアの計算には，分割表 (Contigency Table:ct) を用いる．分割表とは，表 6.1 のような事例サンプルにおいて，異なる事例パターンの頻度を列挙した 5.3 のような表である．

ローカルスコアの計算は，まず，全変数 $\boldsymbol{x}$ に対応する分割表を用いて，変数 $x_i \in \boldsymbol{x}$ に関する条件付き頻度表 cft を 5.4 のように作成し，それを用いてローカルスコア $Score(x_i|\boldsymbol{x} \setminus \{x_i\})$ を計算する．次に，分割表から変数 $x_i \in \boldsymbol{x}$ を周辺消去し，変数集合 $\boldsymbol{w} = \boldsymbol{x} \setminus x_i$ に対応した分割表を作成する．この分割表を用いて，変数 $x_j \in \boldsymbol{w}$ に関する条件付き頻度表を作成し，ローカルスコア $Score(x_j|\boldsymbol{w} \setminus \{x_j\})$ を計算する．この手順を繰り返し全てローカルスコアを計算する．以上のアルゴリズムの疑似コードをアルゴリズム 23 に示す．

表 5.3: 分割表

x_1	x_2	x_3	x_4	頻度
1	0	1	1	3
0	0	0	0	15
1	1	1	1	5
1	1	1	0	1
1	0	1	0	4
1	1	0	0	7
1	0	0	0	11

表 5.4: 条件付き頻度表

x_1	x_3	x_4	$x_2 = 0$	$x_2 = 1$
1	1	1	3	5
0	0	0	15	0
1	1	0	4	1
1	0	0	11	7

アルゴリズム 3　GetLocalScore

- **Input:** 全変数集合 $\boldsymbol{x}$，$\boldsymbol{x}$ に対応する分割表 ct
- **Output:** ローカルスコアの集合 LS

1. **main**

2.　　*getLocalScore(*$\boldsymbol{x}$*, ct)*

3. **end Procedure**

4.

5. **getLocalScore(**$\boldsymbol{w}, ct$**)**

6.　　**for** *all* $x_i \in \boldsymbol{w}$　**do**

7.　　　$cft(ct, x_i)$

8.　　　$LS[x_i][\boldsymbol{w} \setminus \{x_i\}] \leftarrow Score(x_i|\boldsymbol{w} \setminus \{x_i\})$

9.　　**end for**

10.　　**if** $|\boldsymbol{w}| > 1$

11.　　　$getLocalScore(\boldsymbol{w} \setminus \{x_i\}, Ct2Ct(ct, x_i))$

12.　　**end if**

13. **end Procedure**

ここで，$Ct2Ct(ct, x_i)$ は分割表 ct から変数 x_i を周辺化して得られる分割表を表し，$cft(ct, x_i)$ は，ct から生成した x_i に関する条件付き頻度表を表す．スコアには AIC,BIC, MDL や BDe(u) など各ノードごとに分解可能 (decomposable) な全てのスコアを利用できる．

次に，アルゴリズム 3 で得られたローカルスコアを用いて，各ノード x_i ごとに，あり得る全ての親ノード候補 $\boldsymbol{C} \subseteq \boldsymbol{x} \setminus x_i$ を所与としたときの最適な親ノード集合 $\Pi_i^*(\boldsymbol{C})$ を決定する．親ノード候補 $\boldsymbol{C}$ のパターンは，ノード x_i ごとに 2^{N-1} 個存在する．

ここで，親ノード候補 $\boldsymbol{C}$ を所与としたときの，ノード x_i の最適な親ノード集合 $\Pi_i^*(\boldsymbol{C})$ は，

- 親ノード候補 $\boldsymbol{C}$ 自身，あるいは，
- 親ノード候補 $\boldsymbol{C}$ からノードをいくつか除外したノード集合 $\{\boldsymbol{C} \setminus \{c\} | c \in \boldsymbol{C}\}$ の中に存在する最適な親ノード集合 $\Pi_i^*(\boldsymbol{C} \setminus \{c\})$

となる．したがって，最適な親ノード集合 $\Pi_i^*(\boldsymbol{C})$ と，それを所与としたローカルスコア $Score(x_i|\Pi_i^*(\boldsymbol{C}))$ は次式で得られる．

$$Score(x_i|\Pi_i^*(\boldsymbol{C})) = max\left(Score(x_i|\boldsymbol{C}), Score1(x_i|\boldsymbol{C})\right). \tag{5.29}$$

ここで，

$$Score1(x_i|\boldsymbol{C}) = \max_{c \in \boldsymbol{C}} \left(Score(x_i|\Pi_i^*(\boldsymbol{C} \setminus \{c\}))\right). \tag{5.30}$$

式 5.29 では，最適親ノード集合 $\Pi_i^*(\boldsymbol{C})$ を決定するために，ノード候補 $\{\boldsymbol{C} \setminus \{c\} | c \in \boldsymbol{C}\}$ 中の最適親ノード集合 $\Pi_i^*(\boldsymbol{C} \setminus \{c\})$ が求められていないといけない．この条件を満たすために，Silander and Myllymaki(2006) では，**辞書式順序** (lexicographic Order) で式 5.29 を再帰的に適用するアルゴリズムを提案している．疑似コードをアルゴリズム 24 に示す．

アルゴリズム 4　GetBestParents

- **Input:** 全変数集合 $\boldsymbol{x}$，ローカルスコア集合 LS
- **Output:** $bps[][]$:最適親ノード集合

```
1. main
2.     for x_i ∈ x do
3.         getBestParents(x,x_i, LS)
4.     end for
5. end Procedure
6.
7. getBestParents(x, x_i, LS)
```

8. $bps[i] = 2^{|\boldsymbol{x}|-1}$ の変数集合

9. $bss[i] = 2^{|\boldsymbol{x}|-1}$ の局所スコア集合

10. **for** *all* $cs \subset \boldsymbol{x} \setminus \{x_i\}$ *in lexicographic order* **do**

11. $bps[i][cs] \leftarrow cs$

12. $bss[i][cs] \leftarrow LS[x_i][cs]$

13. **for** *all* $cs1 \subset cs$ *which* $|cs \setminus cs1| = 1$ **do**

14. **if** *bss[cs1]* $>$ *bss[cs]*

15. $bps[i][cs] \leftarrow bps[cs1]$

16. $bss[i][cs] \leftarrow bss[cs1]$

17. **end if**

18. **end for**

19. **end for**

20. **end Procedure**

アルゴリズム 24 では，全ての親ノード候補が親変数である場合を初期値とし，逐次的にひとつずつ親ノードを消去してスコアが改善されれば消去していく手法を提案している．

このアルゴリズムが提案されるまで，**トポロジカルオーダー** (topological order) と呼ばれるノード順序を所与として最適なグラフ構造を求め，これを全ての可能なノード順序について計算する手法が主流であった．しかし，このアルゴリズムの特徴は，最適なノード順序を以下のように求める．

すなわち，その特徴は，全変数集合 $\boldsymbol{x}$ の全ての部分集合 $\boldsymbol{w} \subseteq \boldsymbol{x}$ について，そのノード集合から構成される最適なサブネットワーク構造と，その最適なリーフノード（“*Sink*” と呼ばれ，外向きのエッジを一つも持たないノードを指す.）を選択することにより，最適なノード順序を求めるのである．

ここで，ノード集合 $\boldsymbol{w}$ からなる最適なネットワーク $G^*(\boldsymbol{w})$ から，任意のリーフノード x_i に関するサブネットワーク $(x_i, \Pi_i^*(\boldsymbol{w} \setminus \{x_i\}))$ を取り除いたサブネットワークを G^{-i} とする．このとき，サブネットワーク G^{-i} は，ノード集合 $\boldsymbol{w} \setminus \{x_i\}$ からなる最適なサブネットワークとなる．なぜなら，$\boldsymbol{w} \setminus \{x_i\}$ で構成される G^{-i} より最適なネットワーク $G^\dagger$ が存在するならば，$G^\dagger$ にサブネットワーク $(x_i, \Pi_i^*(\boldsymbol{w} \setminus \{x_i\}))$ を加えた構造が最適なネットワーク $G^*(\boldsymbol{w})$ として得られているはずであり，矛盾するからである．

この性質を利用し，動的計画法では，少数のノード集合 $\boldsymbol{w}$ からなる最適なサブネットワーク $G^*(\boldsymbol{w})$ に，最適なリーフノード x_i を一つずつ追加することで，全ノード $\boldsymbol{x}$ に対する最適ネットワーク構造 $G^*(\boldsymbol{x})$ を決定する．

ここで，ノード集合 $\boldsymbol{w} \subseteq \boldsymbol{x}$ からなる最適なサブネットワーク構造 $G^*(\boldsymbol{w})$ は次の二つのサブネットワークの和で表される．

- リーフノード x_i とその最適親ノード集合 $\Pi_i^*(\boldsymbol{w} \setminus \{x_i\})$ からなるサブネットワーク $(x_i, \Pi_i^*(\boldsymbol{w} \setminus \{x_i\}))$.
- ノード集合 $\boldsymbol{w} \setminus \{x_i\}$ で構成される最適サブネットワーク.

すなわち，最適なサブネットワーク $G^*(\boldsymbol{w})$ における最適なリーフノードは次式で得られる．

$$\begin{aligned} Leaf^*(\boldsymbol{w}) &= \underset{x_i \in \boldsymbol{w}}{\operatorname{argmax}}\, Skore(\boldsymbol{w}, x_i) \\ &= \underset{x_i \in \boldsymbol{w}}{\operatorname{argmax}}\, (Score(x_i | \Pi_i^*(\boldsymbol{w} \setminus \{x_i\})) + Score(G^*(\boldsymbol{w} \setminus \{x_i\}))) . \quad (5.31) \end{aligned}$$

上式を用いて，あり得る全てのノード集合 $\boldsymbol{w} \subseteq \boldsymbol{x}$ について，最適なサブネットワークとリーフノードを決定する方法はアルゴリズム 25 に示した．まず，リーフノード $x_i \in \boldsymbol{w}$ を選択する．$\boldsymbol{w} \setminus x_i$ から構築される最適サブネットワークのスコア $Score(G^*(\boldsymbol{w} \setminus \{x_i\}))$ と，リーフノード x_i とするサブネットワーク $(x_i, \Pi_i^*(\boldsymbol{w} \setminus x_i))$ のスコア $Score(x_i | \Pi_i^*(\boldsymbol{w} \setminus \{x_i\}))$ の和を計算する．これが，$Skore(\boldsymbol{w}, \cdot)$ より大きければ，$\boldsymbol{w}$ の最適リーフノードを x_i とし，$Skore(\boldsymbol{w}, \cdot)$ の値を更新する．式 5.31 の計算に必要なスコア $Score(x_i | \Pi_i^*(\boldsymbol{w} \setminus \{x_i\}))$ は，アルゴリズム 4 の計算結果を参照することで得られる．一方，スコア $Score(G^*(\boldsymbol{w} \setminus \{x_i\}))$ を得るためには，あらかじめノード集合 $\boldsymbol{w} \setminus \boldsymbol{x_i}$ に対して，式 5.31 を計算しておく必要がある．そこで，Silander and Myllymaki(2006) では，ここでも辞書式順序によるアルゴリズムを提案している．アルゴリズムの疑似コードを以下に示す.

アルゴリズム 5　GetBestLeafs($\boldsymbol{x}, bps, LS$)

- **Input:** $\boldsymbol{x}, bps, LS$
- **Output:** あり得る全てのノード集合を所与としたときの最適リーフノードの集合：$Leafs[]$

1. **for** *all* $\boldsymbol{w} \subseteq \boldsymbol{x}$ *in lexicographic order* **do**

2. $score[\boldsymbol{w}] \leftarrow 0.0$

3. $Leafs[\boldsymbol{w}] \leftarrow -1$

4. **for** *all* $leaf \in \boldsymbol{w}$ **do**

5. $upvars \leftarrow \boldsymbol{w} \setminus \{leaf\}$

6. $skore \leftarrow score[upvars] + LS[leaf][bps[leaf][upvars]]$

7. **if** $Leafs[\boldsymbol{w}] = -1$ *or* $skore > score[\boldsymbol{w}]$

8. $scores[\boldsymbol{w}] \leftarrow skore$

9. $Leafs[\boldsymbol{w}] \leftarrow leaf$

10. **end if**

11. **end for**

12. **end for**

13. **return** $Leafs[]$

アルゴリズム 25 で，あり得る全ての変数集合に対して最適なリーフノードが決まれば，ただちに最適なノード順序 $\boldsymbol{o}$ が次式で決定できる．

$$o_i^*(V) = Leaf^*(V \setminus \cup_{j=i+1}^{|V|}\{o_j^*(V)\}) \tag{5.32}$$

最適ノード順序を求める擬似コードは, アルゴリズム 26 に示す.

アルゴリズム 6　Leafs2Order($\boldsymbol{x}, Leafs$)

- **Input:**$\boldsymbol{x}, Leafs$
- **Output:** ノード順序 $Order$

1. $Order = |\boldsymbol{x}|$ *array*

2. $left = \boldsymbol{x}$

3. **for** i=$|\boldsymbol{x}|$ *to 1* **do**

4. $Order[i] \leftarrow Leafs[left]$

5. $left \leftarrow left \setminus \{Order[i]\}$

6. **end for**

最後に，最適なベイジアン・ネットワーク構造を次式で決定する．

$$G^*_{o_i^*(V)} = g^*_{o_i^*(V)}(V \setminus \cup_{j=1}^{i-1}\{o_j^*(V)\}) \tag{5.33}$$

結果として, ノード順序を所与としたベイジアン・ネットワーク探索の擬似コードは, アルゴリズム 27 で得られる.

アルゴリズム 7　Ord2net($\boldsymbol{x}, ord, bps$)

- **Input:** $\boldsymbol{x}, Order, bps$
- **Output:** $parents$

1. $parents = |\boldsymbol{x}|$ *variable set*

2. $pedeces \leftarrow \phi$

3. **for**$i = 1$ *to* $|\boldsymbol{x}|$ **do**

4. $parents[i] \leftarrow bps[Order[i]][predecs]$

5. $predecs \leftarrow predecs \cup \{Order[i]\}$

6. **end for**

演習問題 7

以下の”getLocalScore.java”内の関数”calcLS”，”getBestParents.java”内の関数”getBestParents”，”getBestNets.java”内の関数”getKBestNets”をそれぞれ実装し，DP による厳密解アルゴリズムを完成させよ．また，DP による厳密解アルゴリズムを用いて，BDeu スコアを最大にするベイジアンネットワークの構造を学習せよ．さらに，DP による厳密解の構造と，Naive Bayes，TAN のそれぞれに関して，データセット”Solar Flare”の分類精度を比較・考察せよ．
※”getLocalScore.java”，”getBestParents.java”，”getBestNets.java”及びデータセット”Solar Flare”はそれぞれ http://www.ai.lab.uec.ac.jp/実験/からダウンロードできる．

ソースコード 5.1: getLocalScore

```
1  import java.io.BufferedWriter;
2  import java.io.File;
3  import java.io.FileOutputStream;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.io.OutputStreamWriter;
7  import java.io.PrintWriter;
8  import java.math.BigInteger;
9  import java.util.ArrayList;
10 import java.util.HashMap;
11 import java.util.Map;
12 import java.util.concurrent.ForkJoinPool;
13 import java.util.concurrent.RecursiveAction;
14 import org.apache.commons.math3.special.Gamma;
15 public class getLocalScore{
16   private int fANB;
```

```
17  public ArrayList<int[]> mLD; //mLD = exactLearning.LD
18  public double alpha; // equivalent sample size
19  public ArrayList<Double[]> LS; // ローカルスコアリスト. LS.
        get(2)[10101111]は2の親が0,1,3,4,6,8の時のローカルス
        コア
20  public getLocalScore(int j, int database, int fANB,
        ArrayList<int[]> ld, double alpha){
21    this.fANB = fANB;
22    mLD = ld;
23    this.alpha = alpha;
24    initLS();
25    setLocalScores();
26    outputLS(database, j);
27  }
28
29  private void initLS(){
30    LS = new ArrayList<>();
31    for(int i = 0; i < exactLearning.NV - fANB; ++i){
32      LS.add(new Double[(int) Math.pow(2, exactLearning.NV -
            fANB - 1)]);
33    }
34  }
35
36  private void setLocalScores(){
37    for(int i = 0; i < exactLearning.NV - fANB; ++i){
38      System.out.println("i = " + i);
39      ArrayList<Integer> parents = new ArrayList();
40      for(int c = 0; c < i; ++c) parents.add(c);
41      for(int c = i; c < exactLearning.NV - fANB - 1; ++c)
            parents.add(c+1);
42      if(fANB == 1) parents.add(exactLearning.NV - 1);
43      Map<BigInteger, int[]> cft = new HashMap<>();
44      setCft(cft, parents, i);
45      double denominator = 1.0;
46      for(int c = 0; c < parents.size(); ++c){
47        denominator *= exactLearning.NC[parents.get(c)];
48      }
49      int j = (1 << (exactLearning.NV - fANB - 1)) - 1;
50      LS.get(i)[j] = calcLS(cft, i, denominator);
51      ForkJoinPool pool = new ForkJoinPool();
52      pool.invoke(new LocalScoreDP(cft, i, 0, j, denominator
            ));
53    }
54  }
55
56  private class LocalScoreDP extends RecursiveAction {
```

```
57      private Map<BigInteger, int[]> cft;
58      private int child;
59      private int p;
60      private int j;
61      private double denominator;
62      LocalScoreDP(Map<BigInteger, int[]> cft, int child, int p
            , int j, double denominator) {
63        this.cft = cft;
64        this.child = child;
65        this.p = p;
66        this.j = j;
67        this.denominator = denominator;
68      }
69      @Override
70      protected void compute() {
71        if(p >= exactLearning.NV - fANB - 1){
72          return;
73        }
74        LocalScoreDP f1 = new LocalScoreDP(cft, child, p + 1, j
              , denominator);
75        f1.fork();
76        int margi = (p >= child) ? p + 1 : p;
77        Map<BigInteger, int[]> cft2 = new HashMap();
78        for(Map.Entry<BigInteger, int[]> e : cft.entrySet()) {
79          BigInteger key = e.getKey();
80          BigInteger prod1 = product(exactLearning.MNC, p);
81          BigInteger prod2 = prod1.multiply(BigInteger.valueOf(
                exactLearning.MNC));
82          BigInteger down = key.remainder(prod1);
83          BigInteger up = key.divide(prod2);
84          key = up.multiply(prod2).add(down);
85          if(cft2.containsKey(key)){
86            for(int index = 0; index < exactLearning.NC[child];
                  ++index){
87              cft2.get(key)[index] += e.getValue()[index];
88            }
89          }else{
90            int[] t = e.getValue().clone();
91            cft2.put(key, t);
92          }
93        }
94        int j2 = j - (1 << p);
95        denominator /= exactLearning.NC[margi];
96        LS.get(child)[j2] = calcLS(cft2, child, denominator);
97        LocalScoreDP f2 = new LocalScoreDP(cft2, child, p + 1,
              j2, denominator);
```

```
98      f2.invoke();
99      f1.join();
100   }
101 }
102
103 private void outputLS(int database, int cross){
104   try{
105     File file = new File("data/"+exactLearning.db[database
          ]+"/LS-"+exactLearning.md[exactLearning.model]+"/LS
          "+(cross+1)+".txt");
106     if (checkBeforeWritefile(file)){
107       PrintWriter pw = new PrintWriter(new BufferedWriter(
            new FileWriter(file)));
108       PrintWriter p_writer = new PrintWriter(new
            BufferedWriter(new OutputStreamWriter(new
            FileOutputStream(file),"UTF-8")));
109       for(int i = 0; i < exactLearning.NV - fANB; ++i){
110         for(int j = 0; j < (1 << (exactLearning.NV - fANB
              - 1)); ++j){
111           pw.print(LS.get(i)[j]);
112           pw.print(" ");
113         }
114         pw.println();
115       }
116       pw.close();
117     }else{
118       System.out.println("ファイルに書き込めません");
119     }
120   }catch(IOException e){
121     System.out.println(e);
122   }
123 }
124
125 private static boolean checkBeforeWritefile(File file){
126   if (file.exists()){
127     if (file.isFile() && file.canWrite()){
128       return true;
129     }
130   }
131   return false;
132 }
133
134 private void setCft(Map<BigInteger, int[]> cft, ArrayList<
      Integer> parents, int i){
135   for(int l = 0; l < mLD.size(); ++l){
136     BigInteger key = makeKey(mLD.get(l), parents);
```

```
137       int index = mLD.get(l)[i];
138       if(cft.containsKey(key)){
139         cft.get(key)[index]++;
140       }else{
141         int[] t = new int[exactLearning.NC[i]];
142         for(int j = 0; j < exactLearning.NC[i]; ++j){
143           t[j] = 0;
144         }
145         t[index] = 1;
146         cft.put(key, t);
147       }
148     }
149   }
150
151   private double calcLS(Map<BigInteger, int[]> cft, int i,
          double denominator){
152     double L = 0.0;
153     double alpha_ij = alpha;
154     alpha_ij /= denominator;
155     //BDeu
156
157
158
159     return L;
160   }
161
162   private BigInteger makeKey(int[] data, ArrayList<Integer>
          parents){ // 親状態 -> 数値 に変換する
163     if(!parents.isEmpty()){
164       BigInteger key = BigInteger.ZERO;
165       for(int p = 0; p < parents.size(); ++p){
166         int t = parents.get(p);
167         key = key.add(product(exactLearning.MNC, p).multiply(
                BigInteger.valueOf(data[t])));
168       }
169       return key;
170     }else{
171       return BigInteger.ZERO;
172     }
173   }
174
175   public BigInteger product(int a, int b){// a^b
176     BigInteger r = BigInteger.ONE;
177     for(int i = 0; i < b; ++i){
178       r = r.multiply(BigInteger.valueOf(a));
179     }
```

```
180     return r;
181   }
182 }
```

ソースコード 5.2: getBestParents

```
1  import java.io.BufferedReader;
2  import java.io.BufferedWriter;
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.FileNotFoundException;
6  import java.io.FileWriter;
7  import java.io.IOException;
8  import java.io.InputStreamReader;
9  import java.io.PrintWriter;
10 import java.util.ArrayList;
11 import java.util.Comparator;
12 import java.util.PriorityQueue;
13
14 public class getBestParents {
15   private ArrayList<Double[]> LS;
16   private ArrayList<ArrayList<ArrayList<Integer>>> BP;
17   //BP.get(2).get(1011101).get(0) :変数 2の親候補が 0,3,4,5,7
         である計 2^5のパターンのローカルスコアの中で最大のスコ
         ア．最後のget(0)は無意味.全てのBP で get(0)しか使わない.
18   private int I;
19   private int fANB;
20   public getBestParents(int j, int database, int fANB){
21     this.fANB = fANB;
22     inputLS(database, j);
23     initBP();
24     setBestParents();
25     outputBP(database, j);
26   }
27
28   private void inputLS(int database, int cross){
29     LS = new ArrayList<>();
30     try{
31       FileInputStream fis = new FileInputStream("data/"+
             exactLearning.db[database]+"/LS-"+exactLearning.md[
             exactLearning.model]+"/LS"+(cross+1)+".txt");
32       InputStreamReader isr = new InputStreamReader(fis, "UTF
             -8");
33       BufferedReader br = new BufferedReader(isr);
34       String str = null;
35       for(int i = 0; i < exactLearning.NV - fANB; ++i){
```

```
36          str = br.readLine();
37          String[] tt = str.split(" ").clone();
38          Double[] t = new Double[(int)Math.pow(2,
                exactLearning.NV - fANB - 1)];
39          for(int j = 0; j < (int)Math.pow(2, exactLearning.NV
                - fANB - 1); ++j){
40            //System.out.println(i+" "+j);
41            t[j] = Double.parseDouble(tt[j]);
42          }
43          LS.add(t);
44        }
45        br.close();
46      }catch(FileNotFoundException e1){
47        System.out.println(e1);
48      }catch(IOException e1){
49        System.out.println(e1);
50      }
51    }
52
53    private void initBP(){
54      BP = new ArrayList<>();
55      for(int i = 0; i < exactLearning.NV - fANB; ++i){
56        ArrayList<ArrayList<Integer>> t = new ArrayList<
              ArrayList<Integer>>();
57        for(int j = 0; j < Math.pow(2, exactLearning.NV - fANB
              - 1); ++j){
58          t.add(new ArrayList<Integer>());
59        }
60        BP.add(t);
61      }
62    }
63
64    private void setBestParents(){
65      PriorityQueue<int[]> pq = new PriorityQueue<>(10, CCC);
66      for(int i = 0; i < exactLearning.NV - fANB; ++i){
67        I = i;
68        System.out.println("i = " + i);
69        for(int j = 0; j < Math.pow(2, exactLearning.NV - fANB
              - 1); ++j){
70
71
72
73        }
74      }
75    }
76
```

```
77  private void outputBP(int database, int cross){
78    try{
79      File file = new File("data/"+exactLearning.db[database
          ]+"/BP-"+exactLearning.md[exactLearning.model]+"/BP
          "+(cross+1)+".txt");
80      if (checkBeforeWritefile(file)){
81        PrintWriter pw = new PrintWriter(new BufferedWriter(
            new FileWriter(file)));
82        for(int i = 0; i < exactLearning.NV - fANB; ++i){
83          for(int j = 0; j < (int)Math.pow(2, exactLearning.
              NV - fANB - 1); ++j){
84            for(int k = 0; k < BP.get(i).get(j).size(); ++k
                ){
85              pw.print(BP.get(i).get(j).get(k));
86              pw.print(" ");
87              pw.print(LS.get(i)[BP.get(i).get(j).get(k)]);
88              pw.print(" ");
89            }
90            pw.println();
91          }
92        }
93        pw.close();
94      }else{
95        System.out.println("ファイルに書き込めません");
96      }
97    }catch(IOException e){
98      System.out.println(e);
99    }
100 }
101
102 public Comparator<int[]> CCC = new Comparator<int[]>() {
103   @Override
104   public int compare(int[] a, int[] b) {
105     if(LS.get(I)[BP.get(I).get(a[0]).get(a[1])] - LS.get(I
          )[BP.get(I).get(b[0]).get(b[1])] > 0){
106       return -1;
107     }else{
108       return 1;
109     }
110   }
111 };
112
113 private static boolean checkBeforeWritefile(File file){
114   if (file.exists()){
115     if (file.isFile() && file.canWrite()){
116       return true;
```

```
117       }
118     }
119     return false;
120   }
121 }
```

ソースコード 5.3: getBestNets

```
1  import java.io.BufferedReader;
2  import java.io.FileInputStream;
3  import java.io.FileNotFoundException;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.math.BigInteger;
7  import java.util.AbstractMap;
8  import java.util.ArrayList;
9  import java.util.Comparator;
10 import java.util.HashMap;
11 import java.util.PriorityQueue;
12 public class getBestNets {
13   private ArrayList<ArrayList<ArrayList<double[]>>> BP2;
14   private ArrayList<ArrayList<AbstractMap.SimpleEntry<HashMap<
         Integer, ArrayList<Integer>>, Double>>> BN;
15   //BN.get(10111).get(0): 変数 0,1,2,4から構成されるネットワ
         ークで最大のスコアを持つ構造と,その構造のスコアのペア.
         get(0)しか使わない.
16   private int fANB;
17   public getBestNets(int j, int database, int fANB, ArrayList
         <ArrayList<ArrayList<Integer>>> structure){
18     this.fANB = fANB;
19     initBP2();
20     inputBP(database, j);
21     initBN();
22     for(int n = 1; n <= exactLearning.NV - fANB; ++n){
23       setAllKBestNets(exactLearning.NV - fANB, n, 0, 0);
24       delete(exactLearning.NV - fANB, n - 1, 0, 0);
25     }
26     structure.add(getStructure());
27     for(int k = 0; k < 1; ++k){
28       System.out.println("score: "+BN.get((int)Math.pow(2,
             exactLearning.NV - fANB) - 1).get(k).getValue());
29     }
30   }
31
32   private void initBP2(){
33     BP2 = new ArrayList<>();
```

```
34    for(int i = 0; i < exactLearning.NV - fANB; ++i){
35      ArrayList<ArrayList<double[]>> t = new ArrayList<
            ArrayList<double[]>>();
36      for(int j = 0; j < (1 << (exactLearning.NV - fANB -
            1)); ++j){
37        t.add(new ArrayList<double[]>());
38      }
39      BP2.add(t);
40      t = null;
41    }
42  }
43
44  private void inputBP(int database, int cross){
45    try{
46      FileInputStream fis = new FileInputStream("data/"+
            exactLearning.db[database]+"/BP-"+exactLearning.md[
            exactLearning.model]+"/BP"+(cross+1)+".txt");
47      InputStreamReader isr = new InputStreamReader(fis, "UTF
            -8");
48      BufferedReader br = new BufferedReader(isr);
49      String str = null;
50      for(int i = 0; i < exactLearning.NV - fANB; ++i){
51        for(int j = 0; j < (1 << (exactLearning.NV - fANB -
              1)); ++j){
52          str = br.readLine();
53          String[] tt = str.split(" ").clone();
54          for(int k = 0; k < tt.length / 2; ++k){
55            double[] t = new double[2];
56            t[0] = Double.parseDouble(tt[k*2]);
57            t[1] = Double.parseDouble(tt[k*2 + 1]);
58            BP2.get(i).get(j).add(t);
59            t = null;
60          }
61          tt = null;
62        }
63      }
64      br.close();
65    }catch(FileNotFoundException e1){
66      System.out.println(e1);
67    }catch(IOException e1){
68      System.out.println(e1);
69    }
70  }
71
72  private void initBN(){
73    BN = new ArrayList<>();
```

```
74    for(int j = 0; j < (1 << (exactLearning.NV - fANB)); ++
          j){
75      BN.add(new ArrayList<AbstractMap.SimpleEntry<HashMap<
            Integer, ArrayList<Integer>>, Double>>());
76    }
77    AbstractMap.SimpleEntry<HashMap<Integer, ArrayList<Integer
          >>, Double>
78    t = new AbstractMap.SimpleEntry<HashMap<Integer, ArrayList
          <Integer>>, Double>
79    (new HashMap<Integer, ArrayList<Integer>>(), 0.0); //0.0
          で初期化
80    BN.get(0).add(t);
81  }
82
83  private void delete(int size, int n, int j, int k){
84    if(n <= 0){
85      BN.get(j).clear();
86      for(int i = 0; i < exactLearning.NV - fANB; ++i){
87        if((j & (1 << i)) > 0){
88          int m = encode(i, j);
89          BP2.get(i).get(m).clear();
90        }
91      }
92    }else{
93      for(int i = 0; i < size - (n - 1); ++i){
94        delete(size-(i+1), n-1, j + (1 << (k+i)), k+i+1);
95      }
96    }
97  }
98
99  private void setAllKBestNets(int size, int n, int j, int k
        ){
100   if(n <= 0){
101     setKBestNets(j);
102   }else{
103     for(int i = 0; i < size - (n - 1); ++i){
104       setAllKBestNets(size-(i+1), n-1, j + (1 << (k+i)), k
              +i+1);
105     }
106   }
107 }
108
109 private void setKBestNets(int j){
110   PriorityQueue<double[]> pq = new PriorityQueue<>(10, AAA
          );
111
```

```
112
113
114   }
115
116   public ArrayList<ArrayList<Integer>> getStructure(){
117     ArrayList<ArrayList<Integer>> structure = new ArrayList
          ();
118     for(int i = 0; i < exactLearning.NV - fANB; ++i){
119       structure.add(new ArrayList());
120       ArrayList<Integer> t = BN.get((int)Math.pow(2,
            exactLearning.NV - fANB) - 1).get(0).getKey().get(
            i);
121       for(int p = 0; p < t.size(); ++p){
122         structure.get(i).add(t.get(p));
123       }
124       if(fANB == 1){
125         structure.get(i).add(exactLearning.NV - 1);
126       }
127     }
128     if(fANB == 1){
129       structure.add(new ArrayList());
130     }
131     return structure;
132   }
133
134   public Comparator<double[]> AAA = new Comparator<double
        []>() {
135     @Override
136     public int compare(double[] a, double[] b) {
137       if(a[3] - b[3] > 0){
138         return -1;
139       }else{
140         return 1;
141       }
142     }
143   };
144
145   private int encode(int i, int j){
146     int down = j % (1 << i);
147     int up = j / (1 << (i+1));
148     return up*(1 << i) + down;
149   }
150
151   public BigInteger product(int a, int b){// a^b
152     BigInteger r = BigInteger.ONE;
153     for(int i = 0; i < b; ++i){
```

```
154       r = r.multiply(BigInteger.valueOf(a));
155     }
156     return r;
157   }
158 }
```

関連図書

[1] Akaike,H. (1974) A new look at the statistical model identification. *IEEE Transactions on Automatic Control* **19**(6),716-723

[2] Box,G.E.P. and Tiao,G.C. (1992) Bayesian Inference in Statistical Analysis. New York, N.Y., John Wiley and Sons.

[3] Bouckaert,R. (1994) Probabilistic network construction using the minimum description length principle. *Technical Report ruu-cs-94-27*,Utrecht University.

[4] Buntine,W.L. (1991) Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets and P. Bonissone (eds.),*Proc. 7th Conf. Uncertainty in Artificial Intelligence*,52-60.LA,California,Morgan Kaufmann.

[5] Castillo,E., Hadi,A.S., and Solares,C. (1997) Learning and updating of uncertainty in Dirichlet models. *Machine Learning*, **26**,43-63

[6] Chickering,D.M. (1996) Learning Bayesian networks is NP-complete. In D.Fisher and H.Lenz (eds.),*Proc. International Workshop on Artificial Intelligence and Statistics*,121-130

[7] Chickering,D.M. and Heckerman,D. (2000) A comparison of scientific and engineering criteria for Bayesian model selection. *Statistics and Computing*, **10**, 55-62

[8] Cooper,G.F. and Herskovits,E. (1992) A Bayesian Methods for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309-347

[9] Darwiche, A. (2009) Modeling and reasoning with Bayesian networks, Cambridge University Press

[10] Dawid, A.P. (1979) Conditional Independence in Statistical Theory, Journal of the Royal Statistical Society, Series B, 41, 1-33

[11] Dawid, A.P. (1980) Conditional Independence for Statistical Operations, Annals of Statistics, 8, 598-617

[12] Heckerman,D. , Geiger,2012/10/24, and Chickering,D. (1995) Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning* **20**,197-243

[13] Rissanen,J. (1978) Modeling by shortest data description, *Automatica* **14**, 465-471

[14] Koivisto,M. and Sood,K. (2004) Exact Bayesian structure discovery in Bayesian networks, *Journal of Machine Learning Research*, **5**, 549-573.

[15] Malone,B.M., Yuan,C., Hansen,E.A. , Bridges,S. (2011) Improving the Scalability of Optimal Bayesian Network Learning with External-Memory Frontier Breadth-First Branch and Bound Search. In A. Preffer and F.G. Cozman (eds.) *Proc. the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 479-488

[16] Schwarz,G.E. (1978) Estimating the dimension of a model, *Annals of Statistics* **6**(2), 461-464.

[17] Silander,T. and Myllymaki,P. (2006) A simple approach for finding the globally optimal Bayesian network structure. In R. Dechter and T. Richardson(eds.), *Proc. 22nd Conf. Uncertainty in Artificial Intelligence*, 445-452

[18] Silander,T., Kontkanen,P., and Myllymaki,P. (2007) On sensitivity of the MAP Bayesian network structure to the equipment sample size parameter. In K.B. Laskey, S.M. Mahoney, J.Goldsmith (eds.), *Proc. 23d Conf. Uncertainty in Artificial Intelligence*, 360-367

[19] Steck,H. and Jaakkola,T.S. (2002). On the Dirichlet Prior and Bayesian Regularization. In S.Becker, S.Thrun, K.Obermayer(eds.),*Advances in Neural Information Processing Systems (NIPS)*,697-704

[20] Steck,H. (2008) Learning the Bayesian network structure: Dirichlet Prior versus Data. In D.A. McAllester and P.Myllymaki (eds.),*Proc. 24th Conf. Uncertainty in Artificial Intelligence*, 511-518

[21] Suzuki,J. (1993) A Construction of Bayesian networks from Databases on an MDL Principle. In Heckerman, E.H.Mamdani (eds.),*Proc. 9th Conf. Uncertainty in Artificial Intelligence*, 266-273

[22] Suzuki, J. (2006) On strong consistency of model selection in classification. *IEEE Transactions on Information Theory* **52**(11), 4726-4774

[23] Ueno,M. (2010) Learning networks determined by the ratio of prior and data. In P. Grunwald and P. Spirtes (eds.), *Proc. the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 598-605

[24] Ueno,M. (2011) Robust learning Bayesian networks for prior belief. In A. Preffer and F.G. Cozman (eds.) *Proc. the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, 698-707

[25] Jensen, F.V. and Nielsen, T.D. (2007) Bayesian networks and decision graphs, Springer

[26] Lauritzen, S.L. (1974) Sufficiency, Prediction and Extreme Models, Scandinavian Journal of Statistics, 1, 128-134