機械学習基礎

赤穂 昭太郎

akaho@ism.ac.jp

https://github.com/toddler2009/ml-tutorial

2025 年 4 月 18 日

目次

第 4 章

ベイズモデリング

4.1　ベイズ的アプローチの位置づけと有用性

機械学習におけるベイズモデリングとは，観測データに対して確率的な生成過程（生成モデル）を仮定し，未知のパラメータや潜在変数に対してベイズの定理を用いて事後分布を推論する枠組みである[*1]. (141)

$$\text{事後分布}\quad p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)\, p(\theta)}{p(\mathcal{D})}$$

ここで，$\mathcal{D}$ は観測データ，θ はモデルパラメータ，$p(\mathcal{D} \mid \theta)$ は尤度，$p(\theta)$ は事前分布，$p(\mathcal{D})$ は周辺尤度（証拠）である.

ベイズ的手法は以下のような有用な性質を持つ：

- **不確実性の定量化**：パラメータ推定だけでなく，予測分布を通じて予測の不確実性を表現できる.
- **事前情報の活用**：既存の知識や仮定を事前分布として組み込むことができる.
- **小規模データへの対応**：データ数が少ない状況でも頑健な推定が可能.
- **過学習の抑制**：事前分布や積分的推論によってモデルの複雑さが自動的に調整される.

また，ベイズモデリングは単なるパラメータ推定にとどまらず，モデリング，推論，意思決定を統一的に扱う機械学習の強力な枠組みである.

[*1] ISL では識別問題の一部においてのみ生成モデルを扱っている.

4.1.1　補足：ベイズモデリングにおける定式化と推論の分離

ベイズモデリングでは，パラメータ θ からデータ $\mathcal{D}$ を生成する過程が，しばしば因果的に解釈可能であり，人間にとってモデル化しやすいという利点がある．

この性質を活かし，まず人間が生成過程を明示的なモデルとして定式化し，その後，ベイズの定理を用いて観測されたデータに基づく事後分布の推定を機械的に（計算機によって）行う，という分業的な枠組みが採られる．すなわち，「モデリング（仮定の記述）」と「推論（確率的計算）」の役割を分離することで，複雑な推論問題の解決が容易になる．

この構図は，たとえば小学生が「鶴亀算」などの個別の解法で文章題を解いていたのに対し，中学生になると方程式という一般的な形式に問題を定式化し，あとは機械的な操作によって解を得られるようになるプロセスに類似している．ベイズモデリングは，こうした「定式化」と「解法」の分離を，確率モデルとベイズ推論において実現しているとみなすことができる．

4.2　ベイズモデリングの課題と展開

一方で，ベイズモデリングには以下のような課題も存在する：

- **解析的計算の困難さ**：事後分布の積分や正規化定数の計算が困難な場合が多い．
- **スケーラビリティ**：大規模データに対する推論の計算コストが高い．
- **モデル設計の自由度の高さ**：柔軟性が高い分，適切なモデル・事前分布の設計が求められる．

これらの課題に対処するために，近年では以下のようなベイズモデリングの技術が発展している：

- **生成モデルの設計**：観測変数と潜在変数を統一的に扱う表現力の高いモデル（例：混合モデル，潜在変数モデル，ベイズ線形回帰，ガウス過程など）．
- **グラフィカルモデル**：複雑な確率モデルを視覚的・構造的に表現する枠組み．推論アルゴリズム（変分推論，ベイズ推論，メッセージパッシング）と密接に

関係.

- **推論アルゴリズムの発展**：マルコフ連鎖モンテカルロ法（MCMC），変分ベイズ法，確率的推論などの近似的な手法によって実用的な推論が可能に.

4.3　ベイズモデリングの一般的な手順

ベイズモデリングは，観測データに対して確率的な生成過程を仮定し，その中で未知のパラメータや潜在変数に対してベイズの定理を適用することによって推論を行う枠組みである.

基本的な手順は以下の 2 段階で構成される：

1. **生成モデルの設計**：観測変数 $\boldsymbol{y}$ と入力 $\boldsymbol{X}$，および未知パラメータ $\boldsymbol{\theta}$ に関して，次のようなモデルを設計する：

$$p(\boldsymbol{y}, \boldsymbol{\theta} \mid \boldsymbol{X}) = p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

ここで，$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})$ は尤度関数，$p(\boldsymbol{\theta})$ は事前分布である.

2. **事後分布の推定**：観測データ $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ に基づいて，未知パラメータの事後分布をベイズの定理により求める：

$$p(\boldsymbol{\theta} \mid \boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{y} \mid \boldsymbol{X})}$$

ここで分母 $p(\boldsymbol{y} \mid \boldsymbol{X})$ は周辺尤度（evidence）であり，

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$$

で与えられる.

4.4　ベイズ線形回帰の導入

ベイズモデリングの簡単な例として，線形回帰におけるパラメータ $\boldsymbol{w}$ を確率的に扱う **ベイズ線形回帰**を考える.

4.4.1　モデルの設定

- 入力 $\boldsymbol{x} \in \mathbb{R}^d$
- 出力 $y \in \mathbb{R}$
- パラメータ $\boldsymbol{w} \in \mathbb{R}^d$
- ノイズ $\epsilon \sim \mathcal{N}(0, \sigma^2)$

線形モデルの生成過程は次のように表現される：

$$y = \boldsymbol{w}^\top \boldsymbol{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

これにより，出力の条件付き分布（尤度）は

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(y \mid \boldsymbol{w}^\top \boldsymbol{x}, \sigma^2)$$

4.4.2　データ全体に対する尤度

N 個の独立なデータ $(\boldsymbol{x}_i, y_i)$ に対して，尤度関数は次のようになる：

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) = \prod_{i=1}^{N} \mathcal{N}(y_i \mid \boldsymbol{w}^\top \boldsymbol{x}_i, \sigma^2) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{X}\boldsymbol{w}, \sigma^2 I)$$

ここで，$\boldsymbol{X} \in \mathbb{R}^{N \times d}$ はデザイン行列で，i 行目が $\boldsymbol{x}_i^\top$ である．

4.4.3　事前分布の設定

パラメータ $\boldsymbol{w}$ に対してゼロ平均・等方性の正規事前分布を仮定する：

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{0}, \tau^2 I)$$

4.4.4　事後分布の導出

ベイズの定理により，事後分布は次のように与えられる：

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) \cdot p(\boldsymbol{w})$$

$$\propto \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 - \frac{1}{2\tau^2}\|\boldsymbol{w}\|^2\right)$$

これはガウス分布に比例する指数型分布であり，次のように事後分布は明示的なガウス分布となる：

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{w}_N, \Sigma_N)$$

ただし，

$$\Sigma_N = \left(\frac{1}{\sigma^2}\boldsymbol{X}^\top \boldsymbol{X} + \frac{1}{\tau^2} I\right)^{-1}, \quad \boldsymbol{w}_N = \frac{1}{\sigma^2}\Sigma_N \boldsymbol{X}^\top \boldsymbol{y}$$

4.4.5　リッジ回帰との関係

最尤推定では，$\boldsymbol{w}$ を固定パラメータと見なして最小二乗法により推定する．一方，ベイズ回帰では事後分布全体が得られる．

事後分布の平均 $\boldsymbol{w}_N$ は，次の最適化問題の解にもなっている：

$$\boldsymbol{w}_N = \arg\min_{\boldsymbol{w}} \left\{\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2\right\}, \quad \lambda = \frac{\sigma^2}{\tau^2}$$

これはまさに **リッジ回帰（L2 正則化付き最小二乗）の解**であり，ベイズ的枠組みにおいてリッジ回帰は事後分布の平均（MAP 推定）に一致することが分かる．

■まとめ　ベイズ線形回帰は，線形モデルの係数ベクトルに確率的な不確実性を導入し，予測とパラメータの両方についての事後分布を提供する．事前分布にガウス分布を仮定した場合，解析的に事後分布を求めることが可能であり，その平均はリッジ回帰の解と一致する．

4.4.6　補足：ベイズ線形回帰と頻度主義的線形回帰の比較：単回帰の場合

線形回帰において，ベイズ的アプローチと頻度主義的アプローチはモデルの枠組みは共通しつつも，推論の立場や区間の解釈に大きな違いがある．本節では，単回帰モデルを例に，信頼区間と信用区間の違いを式とともに比較する．

4.4.7　単回帰モデルの共通の前提

入力 $x \in \mathbb{R}$ に対する出力 $y \in \mathbb{R}$ を次のモデルで表す：

$$y = w_0 + w_1 x + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

デザイン行列 $\boldsymbol{X} \in \mathbb{R}^{n \times 2}$ は $(1, x)$ の列ベクトルの集合，目的変数 $\boldsymbol{y} \in \mathbb{R}^n$ を用いて，モデルは次のように書ける：

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon}, \quad \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}, \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$$

頻度主義的線形回帰における推定と信頼区間

最小二乗法または最尤法により，係数の推定量は：

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

このとき，予測点 x_* における予測値 $\hat{y}_*$ の分布は以下のようになる：

$$\hat{y}_* = \boldsymbol{x}_*^\top \hat{\boldsymbol{w}}, \quad \boldsymbol{x}_* = \begin{pmatrix} 1 \\ x_* \end{pmatrix}$$

95% の信頼区間は，次の形で与えられる：

$$\hat{y}_* \pm t_{n-2,0.975} \cdot \hat{\sigma} \sqrt{\boldsymbol{x}_*^\top (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{x}_*}$$

ただし $\hat{\sigma}^2$ は残差平方和による不偏分散推定量，$t_{n-2,0.975}$ は自由度 $n-2$ の t 分布の上位 2.5% 点である．

ベイズ線形回帰における推論と信用区間

ベイズ線形回帰では，事前分布として

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \tau^2 \boldsymbol{I})$$

を仮定し，観測データ $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ に基づく事後分布は以下のようになる：

$$p(\boldsymbol{w} \mid \boldsymbol{y}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{w}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\Sigma}_N = \left(\frac{1}{\sigma^2} \boldsymbol{X}^\top \boldsymbol{X} + \frac{1}{\tau^2} \boldsymbol{I} \right)^{-1}, \quad \boldsymbol{w}_N = \frac{1}{\sigma^2} \boldsymbol{\Sigma}_N \boldsymbol{X}^\top \boldsymbol{y}$$

このとき，予測点 x_* に対する予測分布は：

$$p(y_* \mid x_*, \mathcal{D}) = \mathcal{N}\left(\boldsymbol{x}_*^\top \boldsymbol{w}_N,\ \sigma^2 + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_* \right)$$

95% の信用区間（credible interval）は，この正規分布の 95% 区間として：

$$\boldsymbol{x}_*^\top \boldsymbol{w}_N \pm 1.96 \cdot \sqrt{\sigma^2 + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*}$$

信頼区間と信用区間の比較

- 頻度主義では，$\boldsymbol{w}$ は固定された「真の値」であり，信頼区間は「反復標本における推定値がこの区間に入る確率が 95%」である．
- ベイズ的には，$\boldsymbol{w}$ は確率変数であり，信用区間は「この事後分布において，$\boldsymbol{w}$ がこの区間にある確率が 95%」である．
- 数式的には類似するが，分布の解釈と不確かさの源泉が異なる．
- ベイズ回帰の予測分布には，モデルパラメータの不確かさも統合されている（$\boldsymbol{\Sigma}_N$ の項）．

Program 4.1　Comparison of regression

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from sklearn.linear_model import LinearRegression
4 from sklearn.preprocessing import PolynomialFeatures
```

```
5  from scipy.stats import t
6  from numpy.linalg import inv
7
8  # データ生成
9  np.random.seed(0)
10 n = 5
11 x = np.linspace(0, 1, n)
12 X = np.vstack([np.ones_like(x), x]).T
13 true_w = np.array([1.0, 2.0])
14 sd = 0.2
15 y = X @ true_w + np.random.normal(0, sd, size=n)
16
17 # 予測用データ
18 x_test = np.linspace(-0.1, 1.1, 200)
19 X_test = np.vstack([np.ones_like(x_test), x_test]).T
20
21 # --------------------------------
22 # 頻度主義的線形回帰（信頼区間）
23 # --------------------------------
24 linreg = LinearRegression().fit(X, y)
25 y_pred = linreg.predict(X_test)
26
27 # 残差標準偏差
28 y_train_pred = linreg.predict(X)
29 residual_std = np.sqrt(np.sum((y - y_train_pred)**2) / (
       n - 2))
30
31 # 信頼区間の計算
32 XTX_inv = inv(X.T @ X)
```

```
tval = t.ppf(0.975, df=n - 2)
conf_int = tval * residual_std * np.sqrt(np.sum((X_test @ XTX_inv) * X_test, axis=1))

# --------------------------------
# ベイズ線形回帰（信用区間）
# --------------------------------
sigma2 = sd ** 2          # 観測ノイズ分散
tau2 = 1.0                # 事前分散

Sigma_inv = (X.T @ X) / sigma2 + np.eye(2) / tau2
Sigma_post = inv(Sigma_inv)
w_post_mean = Sigma_post @ (X.T @ y) / sigma2

y_bayes_mean = X_test @ w_post_mean
y_bayes_std = np.sqrt(sigma2 + np.sum((X_test @ Sigma_post) * X_test, axis=1))
bayes_int = 1.96 * y_bayes_std

# --------------------------------
# 図示
# --------------------------------
plt.figure(figsize=(10, 6))
plt.scatter(x, y, label='Data', color='black')

# 頻度主義の回帰直線と信頼区間
plt.plot(x_test, y_pred, label='Frequentist fit', color='blue')
plt.fill_between(x_test, y_pred - conf_int, y_pred +
```

```
    conf_int,
59                  color='blue', alpha=0.2, label='95%
                        Confidence interval')
60
61  # ベイズ回帰の平均と信用区間
62  plt.plot(x_test, y_bayes_mean, label='Bayesian fit',
        color='red', linestyle='--')
63  plt.fill_between(x_test, y_bayes_mean - bayes_int,
        y_bayes_mean + bayes_int,
64                  color='red', alpha=0.2, label='95%
                        Credible interval')
65
66  plt.title('Frequentist vs Bayesian Linear Regression')
67  plt.xlabel('x')
68  plt.ylabel('y')
69  plt.legend()
70  plt.grid(True)
71  plt.tight_layout()
72  plt.show()
```

4.5　ガウス過程回帰（Gaussian Process Regression）

4.5.1　ガウス過程の定義

ガウス過程（Gaussian Process, GP）とは，任意の有限個の点の集合に対して，その関数値が多変量正規分布に従うような確率過程である [12].

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$$

ここで,

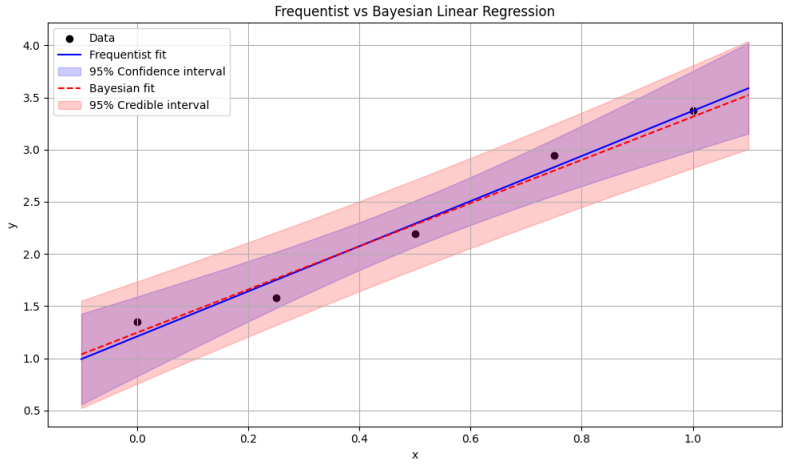

図 4.1　信頼区間とベイズ信用区間の比較

- $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ は平均関数
- $k(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{Cov}(f(\boldsymbol{x}), f(\boldsymbol{x}'))$ は共分散関数（カーネル）

直感的には，ガウス過程は「関数空間上の確率分布」と捉えることができ，関数値を確率的に生成する無限次元の多変量正規分布と見なせる．

4.5.2　ベイズ線形回帰との関係

ベイズ線形回帰では，重みベクトル $\boldsymbol{w}$ にガウス事前分布を置いた．基底関数 $\phi(\boldsymbol{x})$ を使って $f(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$ と表すと，$f(\boldsymbol{x})$ 自体がガウス過程になる：

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}')), \quad k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$$

つまり，ベイズ線形回帰は特定のカーネル（有限次元内積）に対応するガウス過程の特殊例である．

4.5.3　観測データと予測分布

N 個の観測点 $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top$ に対して，対応する出力ベクトルを $\boldsymbol{y} = [y_1, \ldots, y_N]^\top$ とし，ノイズ付きモデルを考える[*2]：

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$$

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

このとき，観測点と新たな入力 $\boldsymbol{x}_*$ における関数値の同時分布は以下のような多変量正規分布になる：

$$\begin{bmatrix} \boldsymbol{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, \boldsymbol{x}_*) \\ K(\boldsymbol{x}_*, X) & K(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix}\right)$$

ここで $K(X,X)$ は観測点間のカーネル行列，$K(X, \boldsymbol{x}_*)$ は観測点と新入力間のカーネルベクトルである．

この共分散構造に基づき，条件付き分布の公式により，f_* の事後分布は次の正規分布として与えられる：

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\mu_*, \sigma_*^2)$$

$$\mu_* = K(\boldsymbol{x}_*, X) \left[K(X,X) + \sigma^2 I\right]^{-1} \boldsymbol{y}$$

$$\sigma_*^2 = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - K(\boldsymbol{x}_*, X) \left[K(X,X) + \sigma^2 I\right]^{-1} K(X, \boldsymbol{x}_*)$$

このようにして，**新たな入力点における予測平均と不確実性（分散）を解析的に計算**することができる．

[*2] ここでは事前分布の平均関数は定数関数 0 としているが，一般に $m_0(x)$ を平均関数とした場合，事後分布の平均関数は以下のものに $m_0(x)$ を単純に足したものとなる．

■ガウス過程回帰の例　人工データでガウス過程回帰をあてはめた例を図 4.2 に示す.

Program 4.2　Gaussian Process

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.gaussian_process import
    GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF,
    WhiteKernel

# ---- 人工データ生成 ----
np.random.seed(2)

# 観測点（密な部分と疎な部分を混在）
x_dense = np.linspace(0.1, 0.3, 3)
x_sparse = np.linspace(0.6, 0.8, 2)
x_train = np.concatenate([x_dense, x_sparse])
y_train = np.sin(2 * np.pi * x_train) + np.random.normal
    (0, 0.1, x_train.shape)

# 入力を次元化（2sklearn の GPR 用）
X_train = x_train.reshape(-1, 1)

# テスト点（予測範囲）
x_test = np.linspace(0, 1, 300).reshape(-1, 1)

# ---- カーネルと GPR モデル定義 ----
kernel = RBF(length_scale=0.2) + WhiteKernel(noise_level
```

```
    =0.1)
gpr = GaussianProcessRegressor(kernel=kernel, alpha=0.0)

# ---- モデル学習と予測 ----
gpr.fit(X_train, y_train)
y_mean, y_std = gpr.predict(x_test, return_std=True)

# ---- 図示 ----
plt.figure(figsize=(10, 6))
plt.plot(x_test, np.sin(2 * np.pi * x_test), 'k--',
    label='True function')
plt.scatter(x_train, y_train, color='black', label='
    Observations')
plt.plot(x_test, y_mean, 'b', label='GPR mean')
plt.fill_between(
    x_test.ravel(),
    y_mean - 1.96 * y_std,
    y_mean + 1.96 * y_std,
    color='blue', alpha=0.2, label='95% Confidence
        interval'
)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Gaussian Process Regression')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

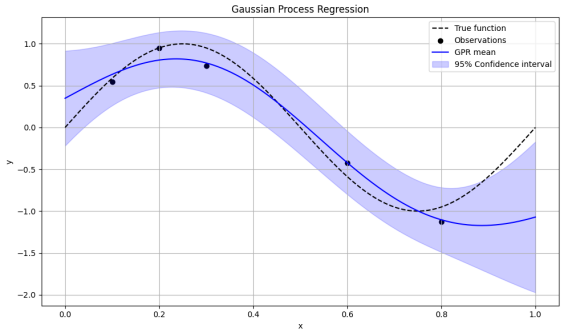

図 4.2　ガウス過程回帰の例

4.5.4　クリギングとの関係

ガウス過程回帰は，空間統計学で用いられる **クリギング（Kriging）** と同一の数理的枠組みである．特に空間補間問題では，測定点の周囲の値を確率的に補間・予測するためにガウス過程が自然に使われる．

4.5.5　ベイズ最適化との関係

ガウス過程回帰は，関数評価のコストが高い最適化問題において，**獲得関数（acquisition function）**を用いたベイズ最適化に応用される．予測分布から得られる平均・分散に基づいて「どこを評価すべきか」を決定できることが，ガウス過程の不確実性モデリングの大きな利点となっている．

4.5.6　補足：ガウス過程と再生核ヒルベルト空間（RKHS）の関係

ガウス過程回帰と RKHS は密接に関係している．特に，ガウス過程の共分散関数 $k(\boldsymbol{x}, \boldsymbol{x}')$ が RKHS の再生核となるとき，ガウス過程の事後平均は RKHS 上の最適化問題の解と一致する．

再生核ヒルベルト空間の定義

RKHS $\mathcal{H}_k$ は，ある正定値対称カーネル関数 $k(\boldsymbol{x}, \boldsymbol{x}')$ に対して定まるヒルベルト空間であり，以下の性質を満たす：

- 各点 $\boldsymbol{x}$ に対して $k(\cdot, \boldsymbol{x}) \in \mathcal{H}_k$
- 任意の $f \in \mathcal{H}_k$ に対して「再生性」が成り立つ：

$$f(\boldsymbol{x}) = \langle f, k(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}_k}$$

この再生性により，関数値が内積で表現できるため，計算上の取り扱いが容易になる．

ガウス過程の平均関数と RKHS の最適化問題

ガウス過程回帰において，MAP 推定値すなわち予測分布の平均 $\mu_*(\boldsymbol{x})$ は，次の変分問題の解と一致することが知られている：

$$f^* = \arg\min_{f \in \mathcal{H}_k} \left\{ \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \right\}$$

この最適化問題は，RKHS ノルムを正則化項とする Tikhonov 正則化であり，カーネルリッジ回帰の変分形式と一致する．

このとき，Representer Theorem により，解 f^* は観測点におけるカーネルの線形結合の形を取る：

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

これはガウス過程回帰における事後平均と同一の形である．

共通点のまとめ

- カーネル k は，ガウス過程の共分散関数でもあり，RKHS の再生核でもある．
- ガウス過程回帰の事後平均（MAP）は，RKHS 上の正則化最小二乗問題の解と一致する．
- ガウス過程回帰は「確率的なカーネル回帰」とも言え，RKHS を背景に持つノンパラメトリックベイズ推論の一例である．

4.6　ハイパーパラメータの推定と周辺尤度最大化

ベイズモデリングにおいて，事前分布やカーネル関数にはハイパーパラメータが含まれる．これらの値はモデルの性能に大きな影響を与えるため，適切に設定する必要がある．

ベイズ的な立場では，ハイパーパラメータ $\boldsymbol{\theta}$ を固定値とせず，観測データ $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ に対する **周辺尤度（marginal likelihood）**：

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \boldsymbol{\theta})\, p(\boldsymbol{w} \mid \boldsymbol{\theta})\, d\boldsymbol{w}$$

を最大化することで, $\boldsymbol{\theta}$ をデータ駆動的に選択することができる．これは **evidence maximization** や 経験ベイズ法 (**empirical Bayes**) とも呼ばれる手法である．

4.6.1　ベイズ線形回帰における周辺尤度

ベイズ線形回帰において，次の仮定を置く：

- 事前分布：$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \tau^2 I)$
- 尤度：$\boldsymbol{y} \mid \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \sigma^2 I)$

このとき，$\boldsymbol{w}$ を積分消去することで，$\boldsymbol{y}$ の周辺分布はガウス分布：

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{0}, \boldsymbol{X}\boldsymbol{X}^\top \tau^2 + \sigma^2 I)$$

よって，対数周辺尤度は次のように明示的に書ける：

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^\top \Sigma^{-1} \boldsymbol{y} - \frac{1}{2}\log\det\Sigma - \frac{n}{2}\log 2\pi, \quad \Sigma = \tau^2 \boldsymbol{X}\boldsymbol{X}^\top + \sigma^2 I$$

ここで，τ^2（事前分布の分散）や σ^2（ノイズ分散）をハイパーパラメータとして最大化する．

4.6.2　ガウス過程回帰における周辺尤度

ガウス過程回帰では，観測モデルとして

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i, \quad f \sim \mathcal{GP}(0, k), \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

を仮定する．カーネル関数 k にもハイパーパラメータ（例：RBF カーネルの幅 ℓ やスケール α）が含まれる．

このとき，観測データ $\boldsymbol{y}$ の周辺分布は：

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{0}, K + \sigma^2 I)$$

ただし $K = K(X, X)$ はカーネル行列であり，ハイパーパラメータに依存する．

対数周辺尤度は以下の通り：

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^\top (K + \sigma^2 I)^{-1} \boldsymbol{y} - \frac{1}{2} \log \det(K + \sigma^2 I) - \frac{n}{2} \log 2\pi$$

これをハイパーパラメータ（カーネルのパラメータ，ノイズ分散など）で最大化することで，最適なモデルを選択できる．

4.6.3　解釈と利点

このような周辺尤度最大化によるハイパーパラメータ推定には以下の特徴がある：

- **モデル選択基準**：データに適したカーネルや正則化の強さを自動的に選択できる．
- **オーバーフィッティングの抑制**：尤度項とモデルの複雑さ（行列式項）のトレードオフが反映される．
- **ベイズ的解釈**：完全ベイズの代替として，事前・事後の積分の一部を解析的に行う方法として整合的.

特にガウス過程回帰では，カーネルの選択が性能に大きく影響するため，周辺尤度最大化は重要なモデル選択手法である．

4.7　ナイーブベイズ法

(154)

ナイーブベイズ法(Naive Bayes classifier)は, 生成モデルに基づく単純な確率的分類器であり, クラスラベル $C \in \{1, \ldots, K\}$ と観測された特徴ベクトル $\boldsymbol{x} = (x_1, \ldots, x_d)$ の同時分布 $p(C, \boldsymbol{x})$ を仮定して推論を行う.

4.7.1　モデルの定義

ベイズの定理に基づく分類では，事後確率を最大化するクラスを選択する：

$$\hat{C} = \arg \max_{c \in \{1,\ldots,K\}} p(C = c \mid \boldsymbol{x}) = \arg \max_{c} \frac{p(\boldsymbol{x} \mid C = c) p(C = c)}{p(\boldsymbol{x})}$$

ここでナイーブベイズ法では，条件付き確率 $p(\boldsymbol{x} \mid C = c)$ に対して，次のような**条件付き独立性の仮定**を置く：

$$p(\boldsymbol{x} \mid C = c) = \prod_{j=1}^{d} p(x_j \mid C = c)$$

これにより，クラスごとの特徴分布を次元ごとに分解し，それぞれ独立にモデル化することができる.

4.7.2　パラメータ推定と分類

学習データから以下を推定する：

- クラスの事前分布 $p(C = c)$（相対頻度など）
- 各特徴のクラス条件付き分布 $p(x_j \mid C = c)$（ベルヌーイ分布，ガウス分布など）

新しいデータ点 $\boldsymbol{x}$ に対しては，以下のルールで分類する：

$$\hat{C} = \arg \max_{c} p(C = c) \prod_{j=1}^{d} p(x_j \mid C = c)$$

4.7.3　長所と短所

■長所

- **学習と推論が非常に高速**：特徴ごとの分布を独立に推定すればよく，計算量が小さい．
- **少ない学習データでも安定**：確率モデルに基づくため，小標本にも比較的強い．
- **理論的に明快**：ベイズ推論の基本形をシンプルに応用．

■短所

- **条件付き独立性の仮定が現実に合わないことが多い**：多くの実データでは特徴量間に相関が存在する．
- **スコアは尤度に比例しており，確率の較正が困難**：確率値は過信すべきでない．
- **連続値の扱いには仮定が必要**：多くの場合，ガウス分布などを仮定する必要がある．

4.7.4　条件付き独立性仮定の破れとその影響

ナイーブベイズ法では，クラス条件付きで特徴量が独立であるという仮定を置くが，これは実データでは通常成立しない．にもかかわらず，ナイーブベイズ法が実践的に有効であることがしばしば観察されている．

この現象に対して，以下のような理論的・経験的知見がある：

- **分類には尤度比が関与する**ため，相関の影響がスコアの順位に現れにくいことがある（Domingos et al[13]）．
- **特徴量の冗長性が高い場合**，相関があっても分類境界にはそれほど影響を与えない．
- **過剰な表現力を避けることで過学習を防いでいる**という観点から，むしろ仮定の単純さが汎化性能に寄与する場合もある．

そのため，ナイーブベイズ法は「モデルは間違っているが予測は正しい」ケースの典型例とされることが多い．より表現力の高いモデル（例：ベイジアンネットワーク，ツリーベースのモデルなど）との比較や，相関構造を部分的に取り入れる拡張も存在する．

4.8 確率的グラフィカルモデルとベイジアンネットワーク

複数の確率変数間の依存構造を明示的に記述し，効率的な表現と推論を可能にする枠組みが **確率的グラフィカルモデル（probabilistic graphical model）**である．

その中でも，変数間の因果的または生成的構造を **有向非巡回グラフ（DAG: Directed Acyclic Graph）** により記述するものは，**ベイジアンネットワーク（Bayesian network）**や，より広く **有向グラフィカルモデル**，**階層ベイズモデル**とも呼ばれる[*3]．

4.8.1 定義と構造

ベイジアンネットワークは，確率変数の集合 $\boldsymbol{X} = (X_1, \ldots, X_n)$ に対して，それぞれの変数の親ノード $\mathrm{Pa}(X_i)$ をグラフ構造で定義し，全体の結合確率分布を以下のように因数分解する（例：図 4.3）：

$$p(\boldsymbol{X}) = \prod_{i=1}^{n} p(X_i \mid \mathrm{Pa}(X_i))$$

この表現は，次のような利点を持つ：

- 条件付き独立性が構造的に組み込まれており，表現の冗長性を削減できる．
- 複雑なドメイン知識（例：因果関係や階層構造）をモデルとして明示的に記述可能．
- 一般の連続変数・離散変数・混合型変数にも対応できる（例：階層ベイズモ

[*3] 世の中にあるツールや使われる文脈によっては確率変数が離散値を取る場合の DAG に限定してベイジアンネットワークと呼ぶこともあるので注意．ここでは確率変数の種類に寄らず一般的な意味でベイジアンネットワークを扱う．

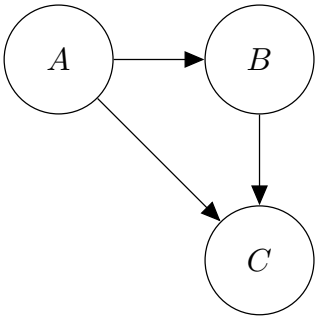

図 4.3　簡単なベイジアンネットワークの例．このベイジアンネットワークは $P(A,B,C) = P(A)P(B \mid A)P(C \mid A,B)$ という分解を与える．ちなみにこれはどんな 3 変数確率分布に対しても成立する式なので分解による制約にはなっていないことに注意.

デル）.

図 4.4,4.5,4.6 にそれぞれナイーブベイズモデル，ベイズ線形回帰，医療診断応用を想定したものに対するベイジアンネットワークを示す.

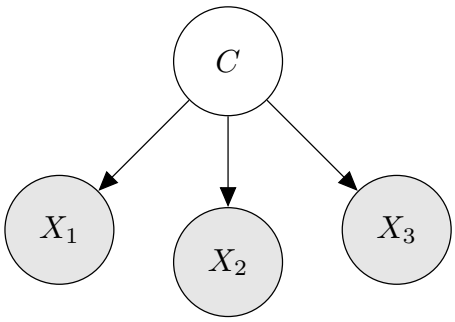

図 4.4　ナイーブベイズモデルのベイジアンネットワーク（C：クラス変数、X_i：条件付き独立な特徴変数）

4.8.2　高次元の問題への対応と注意点

グラフィカルモデルは，条件付き独立性の構造を利用することで，**高次元の結合確率分布を効率的に表現・推定**できるという大きなメリットがある.

例えば，各変数が少数の親ノードにのみ依存する場合，必要なパラメータ数は指数的から線形または多項式オーダーに削減できる.

■次元の呪いの回避

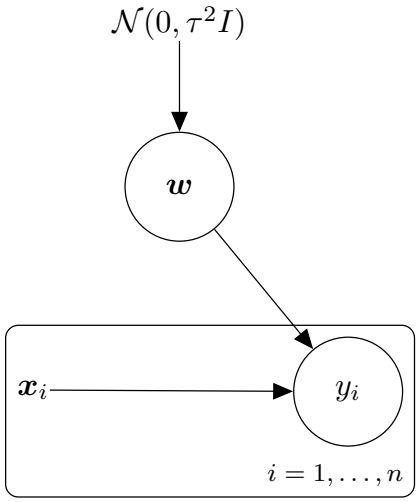

図 4.5　ベイズ線形回帰のベイジアンネットワーク：パラメータ $\boldsymbol{w}$ に対して n 個の観測 $(\boldsymbol{x}_i, y_i)$ が与えられる

- 高次元空間での直接的なモデル化は困難（次元の呪い）.
- 条件付き独立性を利用すれば，変数集合を小さな条件付き分布に分解でき，計算や推定が容易になる.
- これにより，少ないデータでも意味のあるモデル化が可能になる.

■モデルの複雑さと限界　ただし，以下のような問題も存在する：

- **親ノードの数が多くなると条件付き分布の次元が上がる**ため，利点が薄れる.
- 状態数が多い変数が親ノードにあると，指数的な条件付き確率表が必要になることがある.
- モデルの構造が固定でない場合，構造学習の問題も加わり計算が難しくなる.

4.8.3　推論アルゴリズムの概略

ベイジアンネットワークにおける推論（周辺化・条件付き分布の計算）には，構造に依存して様々なアルゴリズムが用いられる：

- **ループのない構造（木やポリツリー）**では，変数消去（variable elimination）やメッセージパッシング（belief propagation）により効率的に推論可能.

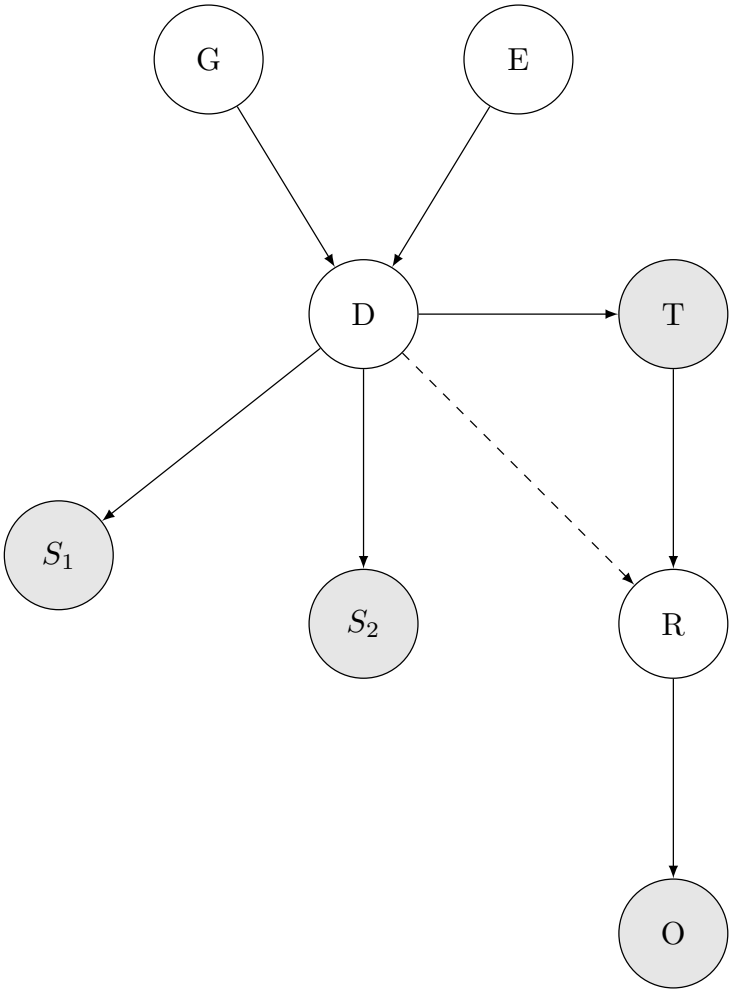

図 4.6　病気診断におけるベイジアンネットワークの例：G（遺伝因子）と E（環境因子）が病名 D に影響し、D が症状 S_1, S_2 検査結果 T、そして治療 R を通じて転帰（Outcome）O に影響を与える。破線は $D \to R$ の直接効果（例えば重症度）を示す。

- **ループを含む構造（一般の DAG）**では，厳密な推論は指数時間を要するが，変分推論や MCMC（後述）といった近似推論法が活用される．

このように，ベイジアンネットワークは構造的な知識を活用しながら，柔軟かつスケーラブルに複雑な確率分布を扱うための基本的な枠組みを提供する．

補足：ベイジアンネットワークにおける「ループ」の異なる意味

ベイジアンネットワークにおいて「ループ（loop）」という用語は，文脈によって異なる意味を持つため，明確な区別が必要である．

■1. 有向閉路（directed cycle）としてのループ　これはグラフ理論的な意味でのループであり，次のような有向辺の列が存在する場合を指す：

$$X_1 \to X_2 \to \cdots \to X_k \to X_1$$

このような **有向閉路（サイクル）** が存在する場合，グラフは **有向非巡回グラフ（DAG）** ではなくなり，ベイジアンネットワークとしては **well-defined な結合分布を構成できない**．よって，ベイジアンネットワークの構造は常に DAG である必要がある．

■2. 木でないグラフにおける「情報のループ」　一方で，推論アルゴリズムやネットワーク構造の議論において「ループがある」と言った場合，それは以下のような構造を指すことが多い：

- グラフが **木（tree）構造**や **ポリツリー（polytree）構造**でない[*4]．
- すなわち，あるノード間に **複数の無向経路（undirected paths）が存在する**．

このような構造では，有向閉路は存在しないものの，無向的な意味で「ループ（閉路）」があるため，推論アルゴリズムにおいては次のような影響が生じる：

- 厳密な推論（例：変数消去法，Belief Propagation）は指数的な計算量となる可能性がある．
- Belief Propagation はポリツリーでは厳密解だが，ループがある場合は近似になる（Loopy Belief Propagation）．

■まとめ

- ベイジアンネットワークは構造として **有向非巡回グラフ（DAG）** である必要がある．
- 推論において「ループがある」と言うときは，無向的に閉路が存在する木構造でないグラフを指す．

[*4] polytree とは枝の向きをなくしたときに木構造とみなせる構造を指す．

- この意味でのループは，厳密推論の困難さや近似アルゴリズムの必要性をもたらす．

この区別を明確にすることで，グラフ構造・推論アルゴリズム・理論的性質の理解が混乱なく行えるようになる．

4.9　因子グラフと推論アルゴリズムの一般化

Belief Propagation は，ベイジアンネットワークやマルコフランダムフィールドのようなグラフィカルモデルにおいて，木構造であれば効率的に周辺分布を計算できる．

この手法をさらに一般化したのが，**因子グラフ（factor graph）** に基づく **Sum-Product アルゴリズム**である．

4.9.1　因子グラフと Sum-Product アルゴリズム

因子グラフとは，変数ノードと因子ノードの 2 種類のノードから構成される **2 部グラフ（bipartite graph）**である（図 4.7）．

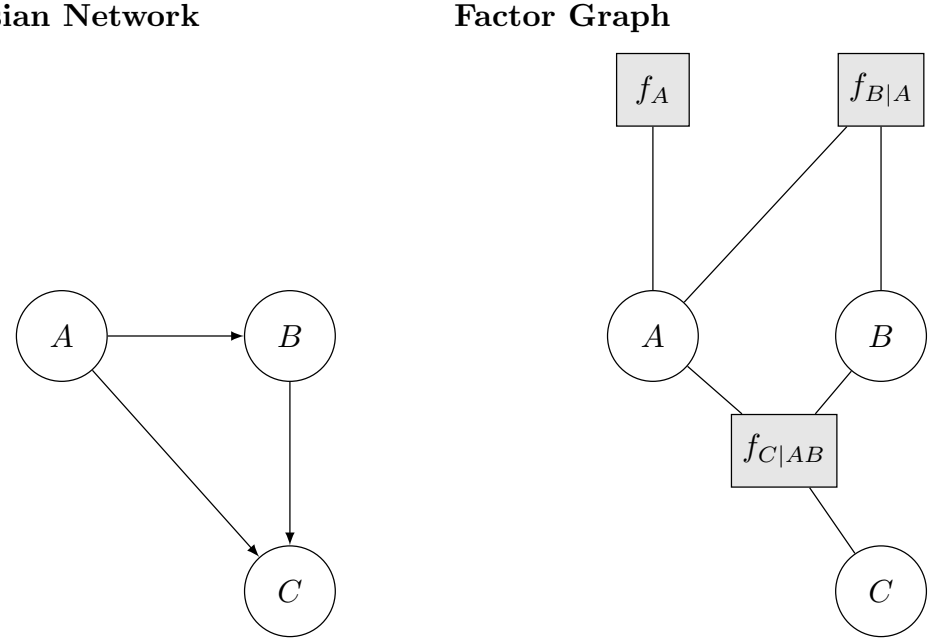

図 4.7　ベイジアンネットワークと対応するファクターグラフの比較：$A \to B \to C$ と $A \to C$ に対応する分解 $f_A \cdot f_{B|A} \cdot f_{C|AB}$

■**定義**　与えられた変数集合 $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ に対して，確率分布が以下のように因子に分解されているとする：

$$p(\boldsymbol{X}) = \frac{1}{Z} \prod_a f_a(\boldsymbol{X}_a)$$

ここで f_a は因子（局所的な関数），$\boldsymbol{X}_a$ は因子 f_a が依存する変数の集合，Z は正規化定数である．

この構造を因子グラフでは次のように表現する：

- 各変数 X_i を変数ノードとして表現
- 各因子 f_a を因子ノードとして表現
- f_a が変数 X_i に依存する場合，X_i と f_a の間にエッジを引く

■Sum-Product アルゴリズム（SP アルゴリズム）　因子グラフにおける推論（周辺分布の計算）は，**Sum-Product アルゴリズム**によって効率的に行える．これは Belief Propagation の因子グラフ版とも言える．

メッセージの送信規則は次の通り：

- 変数ノード i から因子ノード a へのメッセージ：

$$m_{i \to a}(x_i) = \prod_{b \in \mathrm{nb}(i) \setminus \{a\}} m_{b \to i}(x_i)$$

- 因子ノード a から変数ノード i へのメッセージ：

$$m_{a \to i}(x_i) = \sum_{\boldsymbol{x}_a \setminus x_i} f_a(\boldsymbol{x}_a) \prod_{j \in \mathrm{nb}(a) \setminus \{i\}} m_{j \to a}(x_j)$$

ここで $\mathrm{nb}(i)$ はノード i の隣接因子ノードの集合を，$\mathrm{nb}(a)$ は因子ノード a に隣接する変数ノードの集合を表す．

すべてのメッセージが更新された後，各変数の周辺分布は次のように計算される：

$$b_i(x_i) \propto \prod_{a \in \mathrm{nb}(i)} m_{a \to i}(x_i)$$

このアルゴリズムは木構造またはループのない因子グラフであれば，厳密な周辺分布を効率的に計算できる．

4.9.2　Junction Tree アルゴリズム：ループを含む場合の厳密推論

ループがある場合には，Belief Propagation や Sum-Product アルゴリズムは近似になる．ループを含むグラフに対しても厳密推論を可能にするのが，**Junction Tree アルゴリズム（結合木アルゴリズム）**である．

■**基本的なアイデア**　元のグラフから次の変換を行う：

1. グラフの **モラル化（moralization）**：有向グラフの場合，すべての親ノード間に無向辺を追加し，有向辺をすべて無向に変換.
2. **トリアングレーション（chordal 化）**：ループが長い部分を分解し，グラフを「三角化」してクリーク構造に変換.
3. **クリーク（団）から Junction Tree を構成**：最大クリーク（完全連結部分グラフ）をノードとする木構造を作成し，**クリーク間の共有変数（separator）**を通じてメッセージを送る.

■**アルゴリズムの流れ**

- 各クリークに確率因子を割り当てて初期化.
- Junction Tree 上で，Sum-Product のようなメッセージパッシングを行い，すべてのクリークが整合的な周辺分布を持つようにする.
- クリーク内での周辺分布を用いて，任意の変数の周辺分布や条件付き分布を計算.

■**計算コスト**　Junction Tree における計算量は，**最大クリークサイズに指数的**に依存する．したがって，実用上は treewidth が小さいグラフでのみ現実的な方法である.

■**まとめ**

- Sum-Product アルゴリズムは因子グラフ上の BP として統一的に扱える.
- ループのある場合でも，Junction Tree に変換することで厳密推論が可能.
- 計算量は treewidth に依存し，大規模なループ構造では近似推論が現実的.

4.9.3 補足：Turbo 符号・LDPC 符号の実用通信システムへの応用

Belief Propagation や Sum-Product アルゴリズムは，誤り訂正符号の設計・復号アルゴリズムとして現実の通信システムにおいて広く用いられている．特に，**Turbo 符号**および **LDPC 符号**は，無線通信や衛星通信，Wi-Fi などにおける標準的な誤り訂正方式である．

Turbo 符号の応用

Turbo 符号は 1993 年に提案され，Shannon 限界に迫る性能を持つことが実証された．主に次のような通信規格で採用されている：

- 第 3 世代移動体通信（3G, UMTS / W-CDMA）
- CDMA2000
- 一部の 4G LTE 制御チャネル
- 宇宙通信（NASA の深宇宙探査ミッションなど）

Turbo 符号の復号アルゴリズムは，2 つの畳み込み復号器が交互にメッセージを交換する構造を持ち，これは**ループを持つ因子グラフに対する Belief Propagation** と見なすことができる．

LDPC 符号の応用

LDPC 符号（Low-Density Parity-Check code）は，1962 年に Gallager によって提案されたが，1990 年代以降の計算機性能の向上により実用化が進んだ．以下のような最新の通信規格で広く採用されている：

- 第 5 世代移動体通信（5G NR）
- IEEE 802.11（Wi-Fi 6 / Wi-Fi 7）
- デジタル衛星放送（DVB-S2 / DVB-S2X）
- 光ファイバ通信（10GBASE-T, 100GBASE-R など）
- ストレージインターフェース（NVMe over Fabrics など）

LDPC 符号の復号は，因子グラフ上の変数ノードと検査ノード間でのメッセージ更新により行われ，これは **Sum-Product アルゴリズム**の典型例である．

グラフィカルモデルとの関係と意義

Turbo 符号や LDPC 符号の復号は，いずれも **ループのある因子グラフに対する Loopy Belief Propagation（近似推論）**として解釈できる．

- 実際にはループを含むため厳密な推論ではないが，適切な構造設計により非常に高い性能を実現．
- 推論アルゴリズムの工夫が，シャノン限界に近い誤り訂正性能を支えている．
- 機械学習における確率的推論技術との数学的共通点が多く，理論的・実用的に興味深い接点となっている [*5]．

4.10　変分ベイズ法（Variational Bayes, VB）の導入

複雑なベイズモデルでは，事後分布の厳密な計算は難しい．そのため，**近似推論**の枠組みが用いられる．変分ベイズ法は，事後分布 $p(\boldsymbol{Z} \mid \boldsymbol{X})$ を近似分布 $q(\boldsymbol{Z})$ によって近似し，その分布の最適化により推論を行う手法である．

4.10.1　一般的な定式化

ベイズ推論における中心的な対象は，観測データ $\boldsymbol{X}$ に対する事後分布：

$$p(\boldsymbol{Z} \mid \boldsymbol{X}) = \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{p(\boldsymbol{X})}$$

であるが，周辺尤度 $p(\boldsymbol{X}) = \int p(\boldsymbol{X}, \boldsymbol{Z})\, d\boldsymbol{Z}$ の計算が困難な場合が多い．

変分推論では，近似分布 $q(\boldsymbol{Z})$ を導入し，**変分下限（Evidence Lower Bound: ELBO）**を最大化する：

$$\log p(\boldsymbol{X}) \geq \mathbb{E}_{q(\boldsymbol{Z})}\left[\log \frac{p(\boldsymbol{X}, \boldsymbol{Z})}{q(\boldsymbol{Z})}\right] =: \mathcal{L}(q)$$

[*5] 情報理論との接点や実用面では重要だが，統計モデルとして実際に使う機会は少ないかもしれない．

この下限は次のように Kullback-Leibler 距離を用いて表現できる：

$$\log p(\boldsymbol{X}) = \mathcal{L}(q) + \mathrm{KL}(q(\boldsymbol{Z}) \,\|\, p(\boldsymbol{Z} \mid \boldsymbol{X}))$$

よって，$\mathcal{L}(q)$ を最大化することは，q が $p(\boldsymbol{Z} \mid \boldsymbol{X})$ に近づくことを意味する．

■変分分布の仮定　通常，近似分布 $q(\boldsymbol{Z})$ は計算可能性のために，以下のような独立性仮定（mean-field 分解）を置く：

$$q(\boldsymbol{Z}) = \prod_i q_i(Z_i)$$

この仮定により，各因子 q_i の更新式が解析的に導出可能な場合が多い．

4.10.2 Loopy BP との関係

Belief Propagation（BP）は木構造上の厳密推論アルゴリズムであるが，ループのあるグラフに拡張した **Loopy Belief Propagation** は近似推論となる．

興味深いことに，Loopy BP もまた ELBO の最適化問題における **特定の変分分布族（Bethe 近似）**の最適化とみなすことができ，**変分推論の一種**と解釈される（Yedidia et al.[14]）．

4.10.3 混合分布モデルへの適用と EM との比較

変分ベイズ法は，混合分布モデル（例：混合ガウス分布）のように，**潜在変数モデル**に対して特に有効である．たとえば混合ガウス分布では，以下のような構造になる：

- 潜在変数 $\boldsymbol{Z}$（どの成分に属するかのラベル）
- モデルパラメータ $\boldsymbol{\theta} = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}$

変分分布は次のように分解して仮定する：

$$q(\boldsymbol{Z}, \boldsymbol{\theta}) = q(\boldsymbol{Z})q(\boldsymbol{\theta})$$

そして，変分下限 $\mathcal{L}(q)$ を最大化する反復更新法（coordinate ascent）を行う．これは次のように構造的に **EM アルゴリズムと類似**している：

- **E-step（VB 版）**：$q(\boldsymbol{Z})$ を $q(\boldsymbol{\theta})$ に対して最適化
- **M-step（VB 版）**：$q(\boldsymbol{\theta})$ を $q(\boldsymbol{Z})$ に対して最適化

この構造は，標準的な EM アルゴリズム：

$$\theta^{(t+1)} = \arg\max_{\theta} \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X},\theta^{(t)})} \left[\log p(\boldsymbol{X}, \boldsymbol{Z} \mid \theta)\right]$$

と極めて類似している．ただし，VB ではパラメータ θ を点推定するのではなく，分布 $q(\boldsymbol{\theta})$ を最適化する点が異なる．

■比較のまとめ

- EM は点推定（最尤または MAP），VB は分布推定（近似ベイズ）．
- 両者とも潜在変数を持つ生成モデルに対して反復最適化を行う．
- VB の方が不確実性を表現でき，過学習を抑制する傾向がある．

4.10.4　変分ベイズによるガウス混合分布の推論

観測データ $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^D$ に対して，K 成分のガウス混合モデル（GMM）を考える：

$$p(\boldsymbol{x}_n \mid \boldsymbol{\pi}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\Lambda}_{1:K}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$$

ここで：

- π_k：混合比（$\sum_k \pi_k = 1$）
- $\boldsymbol{\mu}_k$：平均ベクトル
- $\boldsymbol{\Lambda}_k$：精度行列（共分散の逆）

潜在変数 $z_n \in \{1, \ldots, K\}$ を導入し，$z_n = k$ のとき $\boldsymbol{x}_n$ は第 k 成分から生成されるとする．

事前分布の仮定（共役事前分布）

- $\boldsymbol{\pi} \sim \mathrm{Dir}(\boldsymbol{\alpha}_0)$

- $\boldsymbol{\mu}_k \mid \boldsymbol{\Lambda}_k \sim \mathcal{N}(\boldsymbol{m}_0, (\beta_0 \boldsymbol{\Lambda}_k)^{-1})$
- $\boldsymbol{\Lambda}_k \sim \mathrm{Wishart}(W_0, \nu_0)$

変分分布の形式（mean-field 分解）

$$q(\boldsymbol{Z}, \boldsymbol{\pi}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\Lambda}_{1:K}) = \left[\prod_{n=1}^{N} q(z_n)\right] q(\boldsymbol{\pi}) \prod_{k=1}^{K} q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$$

VB-E ステップ：$q(z_n)$ の更新

各データ点 $\boldsymbol{x}_n$ に対して，所属確率（責任度）：

$$r_{nk} := q(z_n = k) \propto \exp\left(\mathbb{E}_q[\log \pi_k] + \mathbb{E}_q[\log \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})]\right)$$

具体的には：

$$\log r_{nk} = \mathbb{E}_q[\log \pi_k] - \frac{D}{2}\log(2\pi) + \frac{1}{2}\mathbb{E}_q[\log\det \boldsymbol{\Lambda}_k] - \frac{1}{2}\mathbb{E}_q[(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Lambda}_k (\boldsymbol{x}_n - \boldsymbol{\mu}_k)]$$

すべての k に対して $\{r_{nk}\}_k$ を計算し，正規化する．

VB-M ステップ：$q(\boldsymbol{\pi})$, $q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$ の更新

まず，責任度の合計：

$$N_k := \sum_{n=1}^{N} r_{nk} \qquad \bar{\boldsymbol{x}}_k := \frac{1}{N_k}\sum_{n=1}^{N} r_{nk}\boldsymbol{x}_n \qquad S_k := \sum_{n=1}^{N} r_{nk}(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_k)(\boldsymbol{x}_n - \bar{\boldsymbol{x}}_k)^\top$$

更新式は以下のようになる：

- 混合比の分布（ディリクレ分布）：

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\alpha}), \quad \alpha_k = \alpha_{0k} + N_k$$

- 平均と精度の結合分布：

$$q(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k \mid \boldsymbol{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \cdot \mathrm{Wishart}(\boldsymbol{\Lambda}_k \mid W_k, \nu_k)$$

ここで：

$$\beta_k = \beta_0 + N_k \quad \nu_k = \nu_0 + N_k$$

$$\boldsymbol{m}_k = \frac{\beta_0 \boldsymbol{m}_0 + N_k \bar{\boldsymbol{x}}_k}{\beta_k} \quad W_k^{-1} = W_0^{-1} + S_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)^\top$$

反復更新と収束判定

以上の更新を E/M ステップで交互に繰り返し，変分下限（ELBO）やパラメータの収束を基準に停止する．

4.11　ラプラス近似（Laplace Approximation）

ラプラス近似は，解析的に扱うのが困難な確率分布を，**多変量ガウス分布によって局所的に近似する方法**である．特にベイズ推論においては，**事後分布**や**周辺尤度**の近似評価によく用いられる．

1. 基本的な定式化

D 次元の変数 $\boldsymbol{\theta} \in \mathbb{R}^D$ に対する非正規化分布 $p(\boldsymbol{\theta}) \propto \tilde{p}(\boldsymbol{\theta})$ が与えられており，その近似を考える．

このとき，ラプラス近似では，$\tilde{p}(\boldsymbol{\theta})$ のモード $\hat{\boldsymbol{\theta}}$ のまわりで 2 次のテイラー展開を行い，正規分布で近似する：

$$\log \tilde{p}(\boldsymbol{\theta}) \approx \log \tilde{p}(\hat{\boldsymbol{\theta}}) - \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{H} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})$$

$$\Rightarrow \quad \tilde{p}(\boldsymbol{\theta}) \approx \tilde{p}(\hat{\boldsymbol{\theta}}) \cdot \exp\left\{ -\frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \boldsymbol{H} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\}$$

ここで $\boldsymbol{H}$ は $\hat{\boldsymbol{\theta}}$ におけるヘッセ行列（負の 2 階微分）：

$$\boldsymbol{H} = -\nabla^2 \log \tilde{p}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}}$$

結果として，$p(\boldsymbol{\theta})$ は次のような正規分布で近似される：

$$p(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \hat{\boldsymbol{\theta}}, \boldsymbol{H}^{-1})$$

2. 周辺尤度（モデル証拠）の近似

ベイズモデル選択では，事後分布だけでなく，周辺尤度（evidence）：

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, d\boldsymbol{\theta}$$

を評価する必要がある．ラプラス近似では，積分の主寄与を与えるモード $\hat{\boldsymbol{\theta}}$ のまわりで近似することで，次の近似式が得られる：

$$\log p(\boldsymbol{x}) \approx \log p(\boldsymbol{x} \mid \hat{\boldsymbol{\theta}}) + \log p(\hat{\boldsymbol{\theta}}) + \frac{D}{2}\log(2\pi) - \frac{1}{2}\log\det \boldsymbol{H}$$

ここでも $\boldsymbol{H}$ は $\log p(\boldsymbol{x}, \boldsymbol{\theta})$ の $\hat{\boldsymbol{\theta}}$ におけるヘッセ行列の負値.

この式は，**BIC（ベイズ情報量規準）の導出**においても基礎となる.

3. ラプラス近似の特徴と限界

- 局所的な 2 次近似のため，**モードのまわりでは良好な近似**を与える.
- 非対称な分布，多峰性のある分布では近似精度が悪くなることがある.
- 解析的な手計算や，確率変数の周辺化が困難な場面での**簡便な近似法**として有用.

■ラプラス近似の特徴

- モード近傍での 2 次近似に基づく.
- 局所的な構造を利用するため，**計算が簡便**で解析的に扱いやすい.
- 事後分布が**単峰で対称に近い**場合には有効.
- 多峰性や強い非対称性がある場合には精度が劣る.

■他の手法との比較

特徴	ラプラス近似	MCMC	変分ベイズ法
近似精度	中（単峰）	高（正確）	中〜高（族依存）
計算コスト	低	高	中
分布の形状への対応	弱い（対称・単峰）	強い（任意）	中程度（族次第）
周辺尤度の計算	可（解析）	難しい（サンプル必要）	可（ELBO）
事後分布の表現	正規近似	サンプル集合	近似分布明示

■選択指針（状況に応じた使い分け）

- **高速な解析的近似が必要**：ラプラス近似
- **精度が重要で複雑な分布に対応したい**：MCMC
- **中庸な精度と効率を両立したい**：変分推論

4.12　乱数生成法の基礎

ベイズ推論や MCMC 法をはじめとする多くの確率的手法では，**特定の分布に従う乱数の生成**が基本的な操作となる．本節では，基本的な乱数生成の方法として，**逆関数法**と**棄却法（受容棄却法）**を紹介する．

4.12.1　逆関数法（inverse transform sampling）

確率密度関数（PDF）$f(x)$ を持つ連続分布において，その累積分布関数（CDF）を $F(x) := \int_{-\infty}^{x} f(t)\,dt$ とする．

■基本原理　一様分布 $U \sim \mathrm{Uniform}(0,1)$ の乱数 u に対して，

$$x = F^{-1}(u)$$

とすれば，x は分布 $f(x)$ に従う乱数となる．

■例：指数分布の乱数生成　指数分布（パラメータ λ）の累積分布関数は $F(x) = 1 - e^{-\lambda x}$．よって，

$$x = -\frac{1}{\lambda}\log(1-u) \quad (\text{または } -\frac{1}{\lambda}\log u)$$

は指数分布に従う乱数を与える.

■注意点 逆関数 F^{-1} が解析的に求まらない分布ではこの方法は使えない.

4.12.2 棄却法（acceptance-rejection method）

逆関数法が使えない場合，**提案分布**（proposal distribution）$q(x)$ とその上限を使って目的分布 $f(x)$ から乱数を生成する方法である.

■基本アルゴリズム 以下の手順で乱数 x を得る：

1. $x \sim q(x)$ からサンプルを生成
2. $u \sim \text{Uniform}(0,1)$ を生成
3. $u < \frac{f(x)}{Mq(x)}$ なら x を受容，そうでなければ棄却（再試行）

ここで $M \geq \sup_x \frac{f(x)}{q(x)}$ は十分に大きな定数.

■性質と注意点

- 提案分布 $q(x)$ は $f(x)$ に近い方が効率が高くなる.
- M が大きすぎると多くのサンプルが棄却され，効率が悪くなる.
- $f(x)$ の正規化定数が不明でも比が計算できれば使用可能.

■例：標準正規分布の近似 標準正規分布に対して，提案分布として一様分布や二項分布などが試されるが，通常は Box-Muller 法や Ziggurat 法などの特殊な手法が使われる（詳細は割愛）.

4.12.3 まとめ

- **逆関数法**は累積分布関数の逆写像が解析的に求まる場合に有効.
- **棄却法**は柔軟であり，複雑な分布にも適用できるが効率に注意が必要.
- いずれも MCMC 法の構成要素（提案分布や受容ステップ）の基本を理解する

ために重要.

4.13　MCMC 法の基本原理と代表的アルゴリズム

複雑な確率分布（特に高次元のベイズ事後分布）からのサンプリングを行うために，**マルコフ連鎖モンテカルロ法（Markov Chain Monte Carlo, MCMC）**が広く用いられている.

MCMC 法は，目的の分布 $p(\boldsymbol{x})$ を定常分布とするマルコフ連鎖を構成し，その連鎖から得られるサンプル列を用いて，事後分布や期待値の近似を行う.

4.13.1　MCMC の基本的な枠組み

MCMC 法の本質は，以下の条件を満たすマルコフ連鎖を設計することにある：

- 定常分布（stationary distribution）が $p(\boldsymbol{x})$ に一致する.
- 十分な反復の後，連鎖の分布が $p(\boldsymbol{x})$ に収束する（エルゴード性）.

これにより，サンプル $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}\}$ を用いて，任意の関数 $f(\boldsymbol{x})$ に対する期待値 $\mathbb{E}_p[f(\boldsymbol{x})]$ を次のように近似できる：

$$\mathbb{E}_p[f(\boldsymbol{x})] \approx \frac{1}{T} \sum_{t=1}^{T} f(\boldsymbol{x}^{(t)})$$

4.13.2　メトロポリス法（Metropolis Algorithm）

提案分布 $q(\boldsymbol{x}' \mid \boldsymbol{x})$ に従って候補点 $\boldsymbol{x}'$ を生成し，以下の確率で現在の状態 $\boldsymbol{x}$ を $\boldsymbol{x}'$ に更新する：

$$\alpha(\boldsymbol{x}, \boldsymbol{x}') = \min\left(1, \frac{p(\boldsymbol{x}')}{p(\boldsymbol{x})}\right)$$

この方法は，**対称な提案分布**（$q(\boldsymbol{x}' \mid \boldsymbol{x}) = q(\boldsymbol{x} \mid \boldsymbol{x}')$）の場合に使える.

4.13.3 メトロポリス-ヘイスティングス法（Metropolis-Hastings Algorithm）

メトロポリス法を非対称な提案分布に拡張したもの．提案分布 $q(\boldsymbol{x}' \mid \boldsymbol{x})$ に基づき，以下の受容確率で状態を更新する：

$$\alpha(\boldsymbol{x}, \boldsymbol{x}') = \min\left(1, \frac{p(\boldsymbol{x}')q(\boldsymbol{x} \mid \boldsymbol{x}')}{p(\boldsymbol{x})q(\boldsymbol{x}' \mid \boldsymbol{x})}\right)$$

このアルゴリズムは，**詳細釣り合い条件（detailed balance）**を満たし，$p(\boldsymbol{x})$ を定常分布とする．

4.13.4 ギブスサンプラー（Gibbs Sampling）

変数を成分ごとに更新する方法．多変量分布 $p(x_1, \ldots, x_d)$ に対して，各成分の条件付き分布 $p(x_i \mid \text{rest})$ から順にサンプリングする：

$$x_1^{(t+1)} \sim p(x_1 \mid x_2^{(t)}, \ldots, x_d^{(t)})$$
$$x_2^{(t+1)} \sim p(x_2 \mid x_1^{(t+1)}, x_3^{(t)}, \ldots) \quad \text{など}$$

この方法は，すべての条件付き分布が簡単にサンプリングできるときに非常に効率的．

4.13.5 ハミルトニアンモンテカルロ法（Hamiltonian Monte Carlo, HMC）

連続値空間で高次元な分布に適した MCMC 手法．状態 $\boldsymbol{x}$ に対して「運動量変数」$\boldsymbol{p}$ を導入し，**ハミルトン力学**に基づく時間発展を行う：

$$H(\boldsymbol{x}, \boldsymbol{p}) = U(\boldsymbol{x}) + K(\boldsymbol{p})$$
$$U(\boldsymbol{x}) = -\log p(\boldsymbol{x}), \quad K(\boldsymbol{p}) = \frac{1}{2}\boldsymbol{p}^\top \boldsymbol{M}^{-1}\boldsymbol{p}$$

時間発展は数値的にシミュレーション（Leapfrog 法）し，その結果を Metropolis-Hastings により受容する．大きな空間でも高速かつ効率よく探索可能.

4.13.6　MCMC の収束性と定理

MCMC 法の理論的背景として，次のような定理がある：

- **定常分布の存在**：連鎖がエルゴード的であれば，定常分布 $p(\boldsymbol{x})$ に収束する.
- **エルゴード定理**：長期平均は期待値に収束する：

$$\frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{x}^{(t)}) \xrightarrow{T\to\infty} \mathbb{E}_p[f(\boldsymbol{x})]$$

- **詳細釣り合い条件（detailed balance）**が成立すれば，$p(\boldsymbol{x})$ は定常分布.

4.13.7　まとめ

- MCMC は，目的分布を定常分布とするマルコフ連鎖からのサンプルを用いた推論手法.
- メトロポリス法，MH 法，Gibbs 法，HMC など様々なアルゴリズムがある.
- いずれも詳細釣り合いとエルゴード性に基づく収束性が理論的に保証される.

補足：詳細つり合い条件（Detailed Balance Condition）

MCMC 法において，マルコフ連鎖の状態遷移確率を $T(\boldsymbol{x} \to \boldsymbol{x}')$ とし，目標とする定常分布を $p(\boldsymbol{x})$ とする.

このとき，**詳細つり合い条件（detailed balance condition）**とは次の式で表される：

$$p(\boldsymbol{x})\,T(\boldsymbol{x} \to \boldsymbol{x}') = p(\boldsymbol{x}')\,T(\boldsymbol{x}' \to \boldsymbol{x}) \quad \text{for all } \boldsymbol{x}, \boldsymbol{x}'$$

これは，状態 $\boldsymbol{x}$ から $\boldsymbol{x}'$ への遷移と，$\boldsymbol{x}'$ から $\boldsymbol{x}$ への遷移が，確率の重み付きでバランスしていることを意味する.

■**意味と役割**　この条件が成立すると，全体として $p(\boldsymbol{x})$ は不変となり，連鎖の定常分布となる．すなわち，

$$\sum_{\boldsymbol{x}} p(\boldsymbol{x})\, T(\boldsymbol{x} \to \boldsymbol{x}') = p(\boldsymbol{x}')$$

が自動的に成り立つ（これを**定常性条件**という）．

■**例：メトロポリス-ヘイスティングス法**　MH 法では，提案分布 $q(\boldsymbol{x}' \mid \boldsymbol{x})$ と受容確率 $\alpha(\boldsymbol{x}, \boldsymbol{x}')$ により遷移確率が定義される：

$$T(\boldsymbol{x} \to \boldsymbol{x}') = q(\boldsymbol{x}' \mid \boldsymbol{x})\alpha(\boldsymbol{x}, \boldsymbol{x}')$$

受容確率：

$$\alpha(\boldsymbol{x}, \boldsymbol{x}') = \min\left(1, \frac{p(\boldsymbol{x}')q(\boldsymbol{x} \mid \boldsymbol{x}')}{p(\boldsymbol{x})q(\boldsymbol{x}' \mid \boldsymbol{x})}\right)$$

とすることで，詳細つり合い条件を満たすように設計されている．

■**まとめ**

- 詳細つり合い条件は，定常分布収束のための十分条件．
- 通常の MCMC 手法（MH 法など）はこの条件を満たすように設計される．
- 定常分布の保証に加え，エルゴード性があれば収束性が理論的に確保される．

4.14　能動学習とベイズ最適化

4.14.1　実験計画法の観点からの位置づけ

機械学習において，学習データは与えられるものではなく，自ら選択できる場合がある．このような状況では，限られたデータ取得コストの中で，どのように効率よく情報を集めるかという問題が生じる．

能動学習（Active Learning）および**ベイズ最適化（Bayesian Optimization）**は，こうした文脈において極めて有効な手法であり，ともに「**データを自ら選びながら学習を進める戦略**」であるという共通点をもつ．

この考え方は，統計学における **実験計画法（Design of Experiments, DOE）** の発展形として位置づけることができる．

1. 古典的な実験計画法とその目的

実験計画法は，限られた実験回数の中で，統計的に効率よく因果効果やパラメータ推定を行うための方法論であり，主に次のような目標をもつ：

- 推定量の分散を最小化する（D 最適設計，A 最適設計など）
- パラメータの識別性や非交絡性を確保する
- 実験コストや制約を考慮した設計を行う

これはあくまで統計モデルの推定精度の向上が主目的である．

2. 機械学習における能動的なデータ取得戦略

一方，機械学習では，モデルの予測精度や意思決定の最適化を目的として，以下のような戦略的データ選択が行われる：

- **能動学習**：識別問題において，予測が不確かなサンプルを優先的にラベル付けしてもらうことで，効率よく分類性能を向上させる．
- **ベイズ最適化**：高価な評価関数を最小化／最大化するために，少ない試行回数で最適値を探索する．

これらは，統計的最適設計の発想をベースにしつつ，**予測モデルの更新と次の観測点の選択を逐次的に繰り返す**という点で，より動的な手法である．

3. 共通構造とベイズ的枠組みの利点

能動学習やベイズ最適化は，どちらも次のような枠組みを共有する：

- 現時点でのモデル（多くは確率モデル）に基づき，将来の観測の「価値」を定量化する．
- その価値に基づいて，次に取得すべきデータ点（クエリ）を選択する．
- 新しいデータに基づき，モデルを更新し，再び次の選択へ．

このような繰り返しは，**逐次的ベイズ更新と意思決定の結合**という視点から，ベイズモデリングと非常に親和性が高い.

4. 概要のまとめ

- **実験計画法**は，限られた実験資源の中での最適な観測設計を目的とした古典的手法.
- **能動学習**は，分類モデルなどにおける不確実性を利用し，情報量の高いデータを優先的に取得する戦略.
- **ベイズ最適化**は，高価な評価関数の最適化において，獲得関数（acquisition function）を用いて次の探索点を選択する戦略.
- いずれも，**データ選択と学習のループ**を通じて，効率的に性能を向上させる方法であり，ベイズ的な考え方と密接に結びついている.

4.14.2　能動学習の基本とプールベースドな枠組み

1. 能動学習の基本概念

能動学習（Active Learning）は，限られたラベル付けコストの下で，**効率的に学習データを取得してモデルの性能を高めるための戦略**である.

教師あり学習の前提として，大量のラベル付きデータが必要とされるが，実際にはラベル付けが高価である場面が多い（例：医学画像診断，自然言語アノテーションなど）.

能動学習では，ラベルのないデータから「今ラベル付けすべきサンプル」をモデルが自ら選択する.

2. プールベースドな能動学習

能動学習の代表的な枠組みのひとつが，**プールベースド（pool-based）能動学習**である.

- 未ラベルのサンプル集合（プール） $\mathcal{U} = \{x_1, \ldots, x_n\}$
- 初期のラベル付き集合 $\mathcal{L}$

- モデルを $\mathcal{L}$ で学習し，$\mathcal{U}$ からラベルを追加するクエリ点 $x^\star$ を選ぶ

このような選択と学習のサイクルを繰り返すことで，効率的に性能を向上させる．

3. 代表的なサンプル選択戦略

以下はプールベースド能動学習で用いられる代表的なクエリ戦略である．

■(1) 不確実性サンプリング（Uncertainty Sampling）　モデルが最も予測に自信を持てないサンプルを選ぶ戦略．

2 クラス分類の場合，予測確率 $P(y = 1 \mid x)$ が 0.5 に近いものを優先する：

$$x^\star = \arg\min_{x \in \mathcal{U}} |P(y = 1 \mid x) - 0.5|$$

他の形式として：

- 最小信頼度（Least Confidence）
- マージン（2 クラスの予測確率差）最小
- エントロピー最大化：$-\sum_y P(y \mid x) \log P(y \mid x)$

■(2) クエリ・バイ・コミッティ（Query by Committee, QBC）　モデルの不確実性を，**異なる学習器の意見の不一致**として捉える戦略．

複数のモデルから構成されるコミッティ $\{h_1, \ldots, h_k\}$ を用いて，予測のばらつきが最大のサンプルを選ぶ．

■(3) Expected Model Change / Error Reduction

- あるサンプルをラベル付けしたときに，**モデルが大きく変化する**，あるいは
- **テスト誤差が大きく減少する**ことが期待される点を選ぶ．

理論的には望ましいが，計算コストが高くなるため，近似手法が用いられる．

■(4) 最後に：代表点サンプリング（Diversity Sampling）　未ラベル集合内で「代表的なサンプル」や「密度の高い領域」を優先的に選ぶことで，全体の分布を広くカバーすることを狙う．

4. Exploration-Exploitation のトレードオフ

能動学習における根本的な問題は，次のようなトレードオフである：

- **Exploitation（活用）**：現在のモデルが不確実と判断する点に集中して精度を高める
- **Exploration（探索）**：未知の領域を積極的に探索し，モデルの偏りを是正する

多くの能動学習アルゴリズムはこのバランスを暗に，あるいは明示的にとっている．

- 不確実性サンプリング：exploitation 寄り
- QBC：exploration の要素も含む
- 組合せ戦略：diversity sampling + uncertainty[*6]

このトレードオフの制御は，学習の効率や汎化性能に大きく影響を与える重要な要素である．

5. ベイズモデルに基づく能動学習

能動学習においては，「どのデータ点がより情報を与えてくれるか」を判断する必要がある．そのためには，現在のモデルが持つ不確実性（予測分布の幅や分散）を定量化する必要があり，**ベイズモデルはこの目的に非常に適している**．

■ベイズ的枠組みの特徴

- モデルのパラメータや予測に対して，確率分布を付与することができる
- 不確実性の大きさに応じた「価値に基づくサンプル選択」が可能になる

以下では，代表的な 2 つのベイズ的手法における能動学習の考え方を紹介する．

[*6] 代表点以外にもより未知の (uncertain) 領域からサンプリングするという考え方を入れてトレードオフのバランスを取る戦略

(1) ガウス過程回帰における能動学習

回帰問題において，ガウス過程（Gaussian Process, GP）モデルを用いると，ある入力 x に対して次のような予測分布が得られる：

$$f(x) \mid \mathcal{D} \sim \mathcal{N}\left(\mu(x), \sigma^2(x)\right)$$

ここで：

- $\mu(x)$：予測平均
- $\sigma^2(x)$：予測分散（＝不確実性）

このとき，能動学習における選択戦略として次のようなものが考えられる：

- **最大分散サンプリング**：$\sigma^2(x)$ が最大となる点を選ぶ

$$x^\star = \arg\max_{x \in \mathcal{U}} \sigma^2(x)$$

- **ベイズ最適化との関係**：目的関数の最小化を狙う場合，獲得関数（acquisition function）に基づいて選択する（例：UCB, EI など (後述)）

(2) 確率的分類器を用いた能動学習

分類問題において，ロジスティック回帰やナイーブベイズ分類器のような**確率的分類器**は，入力 x に対して各クラスに属する確率を出力する：

$$P(y \mid x, \mathcal{D})$$

これを活用して，不確実性サンプリングを以下のように定式化できる：

- **最小信頼度**：

$$x^\star = \arg\min_{x \in \mathcal{U}} \max_{y} P(y \mid x)$$

- **最大エントロピー**：

$$x^\star = \arg\max_{x \in \mathcal{U}} -\sum_{y} P(y \mid x) \log P(y \mid x)$$

■ベイズ分類器を用いた QBC ベイズ的な視点から，**クエリ・バイ・コミッティ（QBC）**を実装することも可能である．例えば：

- モデルの事後分布からサンプリングされた複数の分類器（posterior samples）を構成員とする
- 予測のばらつき（分散や投票の不一致）を元にクエリ点を選ぶ

まとめ

- ベイズモデルは不確実性の定量化を自然に行えるため，能動学習と非常に相性が良い．
- ガウス過程回帰では予測分散が明示的に得られるため，分散最大化戦略がとれる．
- 確率的分類器では，予測確率に基づいてエントロピーや信頼度などの情報指標が利用できる．
- モデルの事後分布から多様な予測を得ることで，QBC 型の戦略もベイズ的に構築可能である．

4.14.3 ベイズ最適化

ベイズ最適化（Bayesian Optimization）は，評価コストが高く，導関数が得られないような関数（ブラックボックス関数）を，できるだけ少ない評価回数で最適化するための戦略である．

$$x^\star = \arg\min_{x \in \mathcal{X}} f(x)$$

ただし，$f(x)$ は明示的な形式を持たず，1 回の評価が非常に高価である（実験，シミュレーション，チューニング等）．

1. ベイズ最適化の基本枠組み

ベイズ最適化は以下のステップで構成される：

1. 事前分布（通常はガウス過程）を用いて，f に関する予測モデルを構築する．
2. 獲得関数（acquisition function）を定義し，それを最大化する点 x_t を選ぶ：

$$x_t = \arg\max_x a_t(x)$$

3. 選ばれた x_t で $f(x_t)$ を評価し，モデルを更新する．
4. 所望の予算に達するまで手順を繰り返す．

このように，モデルの予測と不確実性を利用して，**探索（exploration）と活用（exploitation）**のバランスを取りながら最適化を進める．

Program 4.3　Bayesian Optimization

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from sklearn.gaussian_process import
       GaussianProcessRegressor
4  from sklearn.gaussian_process.kernels import RBF,
       WhiteKernel
5  from scipy.stats import norm
6
7  # 目的関数（非線形）
8  def objective(x):
9      return np.sin(3 * x) + x**2 - 1.5 * x
10
11 # 獲得関数：Expected Improvement (EI)
12 def expected_improvement(x, gp, y_best, xi=0.01):
13     mu, sigma = gp.predict(x.reshape(-1, 1), return_std=
           True)
14     sigma = sigma.reshape(-1, 1)
15     mu = mu.reshape(-1, 1)
16     imp = y_best - mu - xi
17     Z = imp / sigma
```

```
18      ei = imp * norm.cdf(Z) + sigma * norm.pdf(Z)
19      ei[sigma == 0.0] = 0.0
20      return ei.ravel()
21
22  # 設定
23  np.random.seed(2)
24  n_initial = 3
25  n_iter = 3
26  bounds = (0.0, 2.0)
27
28  # 初期点
29  X_obs = np.random.uniform(bounds[0], bounds[1], size=(
        n_initial, 1))
30  y_obs = objective(X_obs).ravel()
31
32  # テスト点（可視化用）
33  X_test = np.linspace(bounds[0], bounds[1], 1000).reshape
        (-1, 1)
34  y_true = objective(X_test)
35
36  # カーネルと GP モデル
37  kernel = RBF(length_scale=0.3) + WhiteKernel(noise_level
        =1e-6)
38
39  # 図の準備（各回で上下段：行212*(n_iter+1)列）
40  fig, axes = plt.subplots(2, n_iter + 1, figsize=(5 * (
        n_iter + 1), 8))
41
42  for i in range(n_iter + 1):
```

```
43      ax1 = axes[0, i]  # 上段：GP の当てはめ
44      ax2 = axes[1, i]  # 下段：獲得関数 () EI
45
46      # GPR fit
47      gp = GaussianProcessRegressor(kernel=kernel, alpha=1
            e-6)
48      gp.fit(X_obs, y_obs)
49      mu, sigma = gp.predict(X_test, return_std=True)
50
51      # EI 計算
52      ei = expected_improvement(X_test, gp, np.min(y_obs))
53
54      # 上段：当てはめ関数と信頼区間
55      ax1.plot(X_test, y_true, 'k--', label="True function
            ")
56      ax1.plot(X_test, mu, 'b-', label="GP mean")
57      ax1.fill_between(X_test.ravel(), mu - 1.96 * sigma,
            mu + 1.96 * sigma,
58                       alpha=0.2, color='blue', label="95%
                             CI")
59      ax1.scatter(X_obs, y_obs, c='black', s=40, label="
            Observations")
60      ax1.set_xlim(bounds)
61      ax1.set_title(f"Iteration {i}")
62      ax1.legend()
63
64      # 次の点を獲得関数から取得し描画
65      if i < n_iter:
66          x_next = X_test[np.argmax(ei)]
```

```
        y_next = objective(x_next)

        ax1.axvline(x_next, color='red', linestyle='--',
            label="Next x")

        # 次の観測を追加
        X_obs = np.vstack([X_obs, [[x_next.item()]]])
        y_obs = np.append(y_obs, y_next)

    # 下段：獲得関数 () EI
    ax2.plot(X_test, ei, 'green', label="Expected
        Improvement")
    ax2.fill_between(X_test.ravel(), 0, ei, color='green
        ', alpha=0.2)
    ax2.set_xlim(bounds)
    ax2.set_ylim(bottom=0)
    ax2.set_title("Acquisition (EI)")
    ax2.set_xlabel("x")
    ax2.legend()

plt.tight_layout()
plt.show()
```

2. 能動学習との違いと共通点

- **共通点**：どちらもベイズモデルに基づき，次に観測すべき点を選ぶという逐次的な戦略.
- **主な違い**：
  - 能動学習：予測モデル全体の精度向上が目的（分類性能など）
  - ベイズ最適化：目的関数の最小値（または最大値）をいかに効率よく見つ

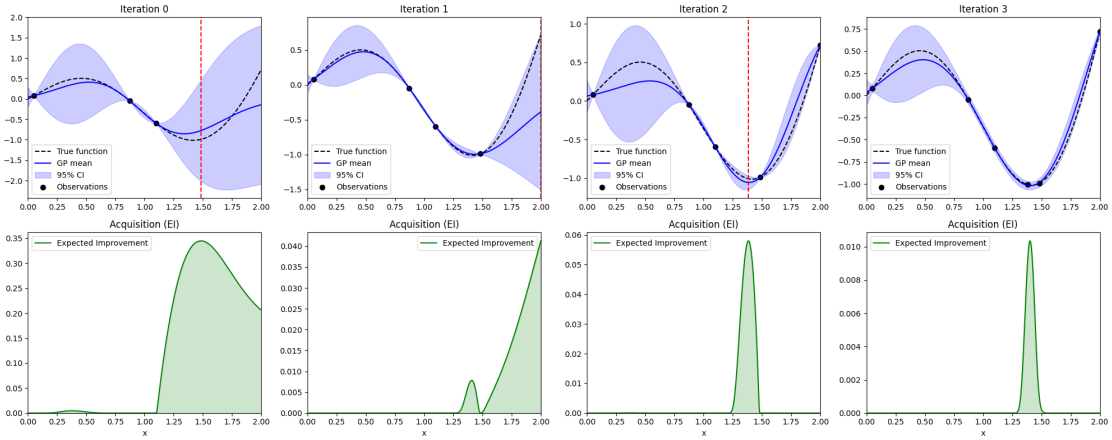

図 4.8　ベイズ最適化 (ここでは関数の最小値を求めている)

けるかが目的

3. 代表的な獲得関数（Acquisition Functions）

ベイズ最適化のコアは，**獲得関数**によって探索点を選ぶ部分である．以下に主要な例を示す．

■(1) 上限信頼境界（Upper Confidence Bound, UCB）

$$a_{\mathrm{UCB}}(x) = \mu(x) - \beta \cdot \sigma(x)$$

あるいは最大化問題では $\mu(x) + \beta \cdot \sigma(x)$．$\beta > 0$ により探索-活用のバランスを調整する．

備考： この形式はマルチアームバンディット問題における UCB 戦略と同型であり，後述の理論解析（後悔最小化など）と関連がある．

■(2) 期待改善量（Expected Improvement, EI）　現在の最良値 f_{best} に対して，改善される期待値：

$$a_{\mathrm{EI}}(x) = \mathbb{E}[\max(f_{\mathrm{best}} - f(x), 0)]$$

ガウス過程予測下では閉形式で計算可能：

$$a_{\mathrm{EI}}(x) = (f_{\mathrm{best}} - \mu(x))\Phi(z) + \sigma(x)\phi(z), \quad z = \frac{f_{\mathrm{best}} - \mu(x)}{\sigma(x)}$$

■(3) 確率的改善（Probability of Improvement, PI）　最良値より良くなる確率：

$$a_{\mathrm{PI}}(x) = \mathbb{P}(f(x) < f_{\mathrm{best}}) = \Phi\left(\frac{f_{\mathrm{best}} - \mu(x)}{\sigma(x)}\right)$$

探索の度合いを操作するためにスケーリングパラメータ ξ を加えることもある．

■(4) 情報利得（Entropy Search, Knowledge Gradient など）　獲得関数を「最小値の位置に関する不確実性の減少量」として定式化する．計算は複雑だが，情報理論的に最も望ましい形式とされる．

4. バンディット問題との関連

UCB のような獲得関数は，バンディット問題（multi-armed bandit）における後悔最小化の枠組みに基づいており，ベイズ最適化もその拡張として捉えることができる．

- バンディット問題：複数の選択肢のうち，報酬が最も大きいものを選ぶ戦略
- ベイズ最適化：連続空間上で同様の「後悔（regret）」を最小にする選択戦略

理論的には，ベイズ最適化においても逐次選択による後悔の上界を示す研究が行われている（例：GP-UCB アルゴリズム）．

まとめ

- ベイズ最適化は，ベイズモデルを用いてブラックボックス関数を効率的に最適化する手法．
- モデルの予測平均と分散を活用し，獲得関数によって探索点を選択する．
- 代表的な獲得関数には，UCB, EI, PI, Entropy Search などがある．
- UCB は特にバンディット問題と関係が深く，後悔最小化の枠組みと接続される．

GP-UCB と後悔境界（Regret Bound）

ベイズ最適化における GP-UCB（Srinivas et al.[15]）は，ガウス過程による予測の平均 $\mu_t(x)$ と分散 $\sigma_t(x)$ を用いて，次のような獲得関数を定義する：

$$x_t = \arg\max_{x \in \mathcal{X}} \left(\mu_{t-1}(x) + \beta_t^{1/2} \cdot \sigma_{t-1}(x) \right)$$

ここで：

- $\mu_{t-1}(x), \sigma_{t-1}(x)$ は時刻 $t-1$ におけるガウス過程の予測平均・標準偏差
- β_t は時間依存の探索係数（理論的な制御パラメータ）

この選択戦略は，**上限信頼区間**を最大化する点を逐次的に評価していく．

■累積後悔（Cumulative Regret）　最適値 $x^\star = \arg\min f(x)$ に対して，時刻 t に選択した点の後悔は：

$$r_t := f(x_t) - f(x^\star)$$

累積後悔は次のように定義される：

$$R_T := \sum_{t=1}^{T} r_t$$

■理論的な上界　適切な条件の下で，GP-UCB は高い確率で次のような後悔境界を満たすことが示されている：

$$R_T = O\left(\sqrt{T \cdot \gamma_T \cdot \beta_T} \right)$$

ここで：

- γ_T は T 回の観測におけるガウス過程の情報ゲイン（information gain）であり，カーネル関数と探索空間に依存する．
- β_T は探索-活用のバランスを調整する係数（通常は対数的に増加）

■情報ゲイン γ_T の例　カーネルが RBF（Gaussian）である場合，$\gamma_T = O((\log T)^{d+1})$ のような増加をする．したがって：

$$R_T = O\left(\sqrt{T \cdot (\log T)^{d+1}} \right)$$

これは，ガウス過程を用いたベイズ最適化が理論的にも **サブ線形の後悔（no-regret）を達成できる** ことを意味する．

まとめ

- GP-UCB は探索（分散）と活用（平均）のバランスをとる戦略．
- カーネルに応じた情報ゲイン γ_T により後悔の速度が決まる．
- 適切なパラメータ設定により，サブ線形後悔 $R_T = o(T)$ が保証される．

参考文献

[1] James, G., Witten, D., Hastie, T., Tibshirani, R. (2023) An Introduction to Statistical Learning, Second Edition, Springer. https://www.statlearning.com

[2] Bishop, C.M. (2026) Pattern recognition and machine learning, Springer (ビショップ：パターン認識と機械学習（上下），丸善)

[3] Kamishima, T., Akaho, S., Asoh, H., Sakuma, J. (2012). Fairness-aware classifier with prejudice remover regularizer. In Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23 (pp. 35-50). Springer Berlin Heidelberg.

[4] 萩原克幸 (2022)，入門 統計的回帰とモデル選択，共立出版.

[5] 赤穂昭太郎 (2008), カーネル多変量解析，岩波書店.

[6] Wahba, G. (1990). Spline models for observational data. Society for industrial and applied mathematics.

[7] 金森敬文, 竹之内高志, 村田昇 (2009). パターン認識 (R で学ぶデータサイエンス 5) 共立出版.

[8] Cover, T., Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

[9] 青嶋誠，矢田和善 (2019)，高次元の統計学，共立出版

[10] 小西貞則. (2010). 多変量解析入門: 線形から非線形へ. 岩波書店.

[11] Hyvärinen, A. (2013). Independent component analysis: recent advances. Philosophical Transactions of the Royal Society A: Mathematical, Physical

and Engineering Sciences, 371(1984), 20110534.

[12] 赤穂昭太郎 (2018). ガウス過程回帰の基礎. システム/制御/情報, 62(10), 390-395.

[13] Domingos, P., Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. Machine learning, 29, 103-130.

[14] Yedidia, J. S., Freeman, W. T., Weiss, Y. (2001). Bethe free energy, Kikuchi approximations, and belief propagation algorithms. Advances in neural information processing systems, 13(24).

[15] Srinivas, N., Krause, A., Kakade, S. M., Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995.

[16] Watanabe, S., Opper, M. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. Journal of machine learning research, 11(12).

[17] Vapnik, V.N. (1998). Statistical Learning Theory, Wiley.