Julia で作って学ぶ計算論的神経科学

山本 拓都

2025 年 5 月 1 日

目次

第 1 章

はじめに

1.1 本書の目的と構成

1.1.1 神経科学における計算論

本書では神経科学における数理モデルを主として取り扱う．初めに神経科学におけるモデルの役割について触れておこう．まず，神経科学の目標は端的に言えば「脳神経系を理解する」ことにある．神経科学に限らず，種々の学問分野においては実験と理論の 2 本柱で，対象とする現象や物質の理解が進められる．ここで実験は調査等も踏まえ実データを取る行為とする．理論の役割は複数あり，実験結果の抽象化および統合，仮説の提供，現象の予測等である (**Blohm2020-vc**).「脳神経系を理解する」ということに関して，その定義は研究者により様々である．ここでは脳の計算処理に関する理論的理解を進めるための 1 つの方法として Marr の 3 レベル (Marr's Three Levels) を紹介する (**Marr1982-wk**)．Marr の 3 レベルは視覚系における計算処理の理解を主としていたが，他でも適用可能である．3 レベルとは (1) 計算理論 (computational theory), (2) 表現・アルゴリズム (representation and algorithm), (3) 実装 (implementation) であり，それぞれの段階での議論や理解を行う．(1) では脳の目的関数とそれを用いた最適化問題の設定を行う．(2) では (1) を実現するための表現およびアルゴリズムを解明する．(3) では (1,2) を神経回路・ハードウェア上で実装する方法を解明することを目標とし，平易には「脳」を作って理解すると言い換えることもできる[*1]．本書ではこの (3) を重視し，読者が自らの手で理論を検証し，数値計算による結果を再現できることを目標とした．また，本書は数式をプログラミングのコードに変換する具体例集としての役割も持っている．モデルの中でも，本書では機械学習に関連する内容が多数登場する．これは神経科学と機械学習は互いに影響を及ぼし合ってきたためである (**Hassabis2017-zm**). 神経科学から機械学習への応用は例えば，ニューラルネットワーク，記憶モデル，注意モデルなどがある．逆に機械学習から神経科学への応用は強化学習，運動制御，ベイズ脳仮説などが挙げられる．筆者の

[*1] ここでの「作る」は計算機等でシミュレーションするという意味であり，脳オルガノイド (brain organoid) を作成するなどの意味ではない

立場としては，神経科学は機械学習の発展のためにあるわけではないので，後者の流れ，すなわち機械学習から神経科学への応用を重視して本書を執筆した次第である．

1.1.2 本書の構成

第 1 章では，Julia 言語の使用法と用いる数学について簡単に説明する．第 2 章から第 5 章までは発火率モデルおよびニューラルネットワークについての説明を行う．第 2 章では，まず神経細胞の簡単な生理学について説明する．発火率モデルを説明したのち，局所学習則によって訓練されるネットワークの説明を行う．第 3 章では，同じく局所学習則ではあるが，ネットワーク全体のエネルギーを下げることを目的としたエネルギーベースモデルと呼ばれる枠組みのネットワークについて説明をする．第 4 章では，誤差逆伝播法に基づいたニューラルネットワークを説明し，貢献度分配問題の生理学的な解決策について説明をする．第 5 章では，さらに再起型ニューラルネットワークを説明し，経時的貢献度分配問題について説明を行う．第 6・7 章ではスパイキングニューラルネットワークとその学習について取り扱う．第 6 章ではネットワークレベルの話から再び神経細胞とシナプスに回帰する．次に，シナプスのダイナミクスについて説明を補いながらモデルを構築する．第 7 章ではネットワークの構築と学習について，第 2 章から第 5 章までを踏まえて説明する．第 8 章から 12 章は上記以外の内容について各論的に説明を行う．第 8 章のリザバーコンピューティングの章では，リザバーコンピューティングと呼ばれる枠組みのネットワークおよびその学習則について，発火率・スパイキングモデルの双方をまとめて紹介する．第 9 章ではベイズ推論の章では，神経回路網により，如何にして確率計算を行うかを説明する．第 10 章では運動学習では，最適制御問題の解決策について説明する．第 11 章の強化学習では，強化学習の基本的事項の説明と，大脳基底核との関連性について説明する．第 12 章は補足的な話題であり，ネットワーク・形態学・グリアについて説明を行う．https://www.sciencedirect.com/science/article/pii/S0364021387800253

1.2 Julia 言語の使用法

1.2.1 Julia 言語の特徴

Julia 言語は本書を執筆するにあたり，なぜ Julia 言語を選択したかというのにはいくつか理由がある．Julia は JIT（Just-In-Time）コンパイルを用いており JIT コンパイラ実行速度が高速であること．ライセンスフリーであり，無料で使用できること．線型代数演算が簡便に書けること．Unicode を使用できるため，疑似コードに近いコードを書けること．他の言語の候補として，MATLAB, Python が挙げられた．MATLAB は神経科学分野で根強く使用される言語であり，線型代数計算の記述が簡便である．なお，線型代数演算の記法に関しては Julia は MATLAB を参考に構築されたため，ほぼ同様に記述することができる．また，MATLAB を使用するには有償ライセンスが必要である．ただし，互換性を持ったフリーソフトウェアである Octave が存在することは明記しておく．Python は機械学習等の豊富なライブラリと書きやすさから広く利用されている言語である．ただし，numpy を用いないと高速な処理を

書けない場合が多く，ナイーブな実装では実行速度が低下してしまう問題がある．線型代数計算も簡便に書くことができず，数式をコードに変換する際の手間が増えるという問題がある．多重ディスパッチ（multiple dispatch）があることは Julia 言語の大きな特徴である．

1.2.2 Julia 言語のインストール方法

Julia (https://julialang.org/) に juliaup (https://github.com/JuliaLang/juliaup) でバージョン管理また，2025 年 3 月以降，Google Colab (https://colab.google/) において Python や R に並んで Julia を選択して使用することが可能となっている．

1.2.3 使用するライブラリ

REPL で] を入力することで，パッケージ管理モードに移行する．本書で使用する Julia ライブラリは以下の通りである．

- IJulia: 開発環境
- PyPlot: 描画用ライブラリ
- LinearAlgebra: 高度な線形代数演算
- Random:

Python では numpy で完結するところをライブラリをいくつも読み込む必要がある点は欠点ではある．描画用のライブラリには `PyPlot.jl` を使用した．`PyPlot` は Python ライブラリである `matplotlib` に依存したライブラリである．Julia で完結させたい場合は `Plot.jl` や `Makie.jl` を使用することが推奨されるが，`PyPlot` (`matplotlib`) の方が高機能であるため，Python がない場合は

```
julia> ENV["PYTHON"] = ""
julia> ]
pkg> build PyCall
```

Python を既にインストールしている場合は，

```
julia> ENV["PYTHON"] = ↵
    raw"C:\Users\TakutoYamamoto\AppData\Local\Programs\Python\Python312\python.exe"
julia> ]
pkg> build PyCall
```

Windows の場合例として Python の実行ファイル (python.exe) への完全なパスを

1.2.4 開発環境

インタプリタ型言語である vscode 筆者は（Python ユーザーでもあるため）Jupyter Lab を使用している．Julia のみで Jupyter Lab を使用するには

```
using IJulia
jupyterlab(detached=true)
```

とすればよい．ただし，この際に Conda を入れることになるため，別途 Python をインストールしておく方が推奨される．p.33 `Pluto.jl` を用いることも可能である

1.2.5 Julia 言語の基本構文

https://docs.julialang.org/en/v1/manual/noteworthy-differences/

1.2.6 命名規則

この節では，本書で用いる Julia の変数名や関数名等に関する基本的な取り決めをまとめる．

変数名

- `nt`: 時間ステップ数 (number of time steps)
- `t`, `tt`: 時間ステップのインデント

1.3 基礎的数学と Julia での記法

本書で使用する数学的内容を整理する．

1.3.1 表記法

本書では次のような記号表記を用いる．

- 実数全体を $\mathbb{R}$, 複素数全体は $\mathbb{C}$ と表記する．
- スカラーは小文字・斜体で x のように表記する．
- ベクトルは小文字・立体・太字で $\mathbf{x}$ のように表記し，列ベクトル (縦ベクトル) として扱う．
- 行列は大文字・立体・太字で $\mathbf{X}$ のように表記する．
- $n \times 1$ の実ベクトルの集合を $\mathbb{R}^n$,$n \times m$ の実行列の集合を $\mathbb{R}^{n \times m}$ と表記する．
- 行列 $\mathbf{X}$ の置換は $\mathbf{X}^\top$ と表記する．ベクトルの要素を表す場合は $\mathbf{x} = (x_1, x_2, \cdots, x_n)^\top$ のように表

記する.

- 単位行列を $\mathbf{I}$ と表記する．$n \times n$ 次元の単位行列は $\mathbf{I}_n$ と表記する.
- ゼロベクトルは $\mathbf{0}$, 要素が全て 1 のベクトルは $\mathbf{1}$ と表記する.
- ベクトル・行列の微分には分子レイアウト記法を使用する.
- 基本的に確率変数は大文字 X のように表記し，確率変数の実現値は小文字 x を用いる．ただし，大文字であっても確率変数でない場合や，実現値がベクトルの場合などがあるため，必ずしもこの規則に従うわけではない.
- e を自然対数の底とし，指数関数を $e^x = \exp(x)$ と表記する．また，自然対数を $\ln(x)$ と表記する.
- 定義を $:=$ を用いて行う．例えば，$f(x) := 2x$ は $f(x)$ という関数を $2x$ として定義するという意味である．定義する対象が右側である場合は，$=:$ を用いる.
- 平均 μ, 標準偏差 σ の正規分布を $\mathcal{N}(\mu, \sigma^2)$ と表記する.

1.3.2 線形代数と微分

線形代数 (Linear Algebra) は、ベクトルや行列といった線形構造を持つ対象の性質を解析する数学の分野であり、現代のあらゆる数学・工学・情報科学の基礎をなしている。線形代数の中心的な対象は、**ベクトル空間**、**線形写像**、およびそれらの表現である**行列**である。まず、**ベクトル空間 (vector space)** とは、スカラー体（通常は実数 $\mathbb{R}$ または複素数 $\mathbb{C}$）に対して定義された加法とスカラー倍という 2 つの演算に関して閉じている集合である。たとえば $\mathbb{R}^n$ は、n 個の実数からなるベクトル全体の集合であり、典型的なベクトル空間の例である。任意のベクトル $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ とスカラー $\alpha \in \mathbb{R}$ に対して、

$$\alpha\mathbf{v} + \mathbf{w} \in \mathbb{R}^n \tag{1.1}$$

が成り立つ。**線形写像 (linear transformation)** とは、ベクトル空間からベクトル空間への写像 $T : V \to W$ であり、加法とスカラー倍に対して線形性を持つもの、すなわち

$$T(\alpha\mathbf{v} + \beta\mathbf{w}) = \alpha T(\mathbf{v}) + \beta T(\mathbf{w}) \tag{1.2}$$

が任意の $\mathbf{v}, \mathbf{w} \in V$ とスカラー α, β に対して成り立つ写像である。このような線形写像は、基底を定めることで**行列 (matrix)** によって表現できる。たとえば、n 次元から m 次元への線形写像は、$m \times n$ の行列 A を用いて

$$\mathbf{y} = A\mathbf{x} \tag{1.3}$$

という形で記述される。ここで $\mathbf{x} \in \mathbb{R}^n$,$\mathbf{y} \in \mathbb{R}^m$ はそれぞれ入力および出力ベクトルである。**行列の積**：$A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$ に対し、$AB \in \mathbb{R}^{m\times p}$ を定義。**転置**：$A^\top$ は行列 A の行と列を交換したもの。**逆行列**：$A \in \mathbb{R}^{n\times n}$ が正則（可逆）であれば、A^{-1} が存在し $AA^{-1} = A^{-1}A = I$ を満たす。**行列式**（determinant）：$\det A$ は正方行列 A に対するスカラー量で、行列の体積のスケーリング率や可逆性の指

標となる。**線形方程式系**の解法である。$A\mathbf{x} = \mathbf{b}$ の形をした方程式において、A の逆行列が存在するならば、その解は

$$\mathbf{x} = A^{-1}\mathbf{b} \tag{1.4}$$

と求められる。**固有値問題**ある正方行列 A に対し、スカラー λ およびベクトル $\mathbf{v} \neq \mathbf{0}$ が

$$A\mathbf{v} = \lambda\mathbf{v} \tag{1.5}$$

を満たすとき、λ は A の**固有値 (eigenvalue)**、$\mathbf{v}$ は**固有ベクトル (eigenvector)** と呼ばれる。固有値分解や対角化は、線形変換の構造解析や行列の関数（例：指数関数）を考える際に中心的な役割を果たす。matrix cookbook に詳しいが,

ベクトル・行列の微分

本書ではベクトルおよび行列の微分を多用する．これは成分ごとに記載するよりも，ベクトル・行列演算をコードに変換しやすいという実装上の利点があるためである．初めに注意したいこととして，ベクトル・行列の微分の記法には分子レイアウト記法 (numerator-layout notation) と分母レイアウト記法 (denominator-layout notation) の 2 種類が存在する．これらは，ベクトル関数やスカラー関数に対する微分の定義の仕方に違いがあり，特に勾配ベクトルの形（行ベクトルか列ベクトルか）や連鎖律の表記に影響を及ぼす．いずれが使用されているかは文献ごとにバラバラであり，中には両方の記法を採用している文献も存在する．本書では，本書では**分子レイアウト記法**を統一的に使用する．記法の例を記述するため，スカラー $x, y \in \mathbb{R}$, ベクトル $\mathbf{x} = [x_i] \in \mathbb{R}^n, \mathbf{y} = [y_j] \in \mathbb{R}^m$, 行列 $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{p\times q}$ を使用する．分子（従属変数）と分母（独立変数）の組み合わせから，次の 6 通りの微分が定義される．

	スカラー	ベクトル	行列
スカラー	$\frac{\partial y}{\partial x}$	$\frac{\partial \mathbf{y}}{\partial x}$	$\frac{\partial \mathbf{A}}{\partial x}$
ベクトル	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	
行列	$\frac{\partial y}{\partial \mathbf{A}}$		

行名が分子の変数の種類，列名が分母の変数の種類を表している．まず，スカラーで偏微分する場合は,

$$\frac{\partial y}{\partial x} \in \mathbb{R}, \quad \frac{\partial \mathbf{y}}{\partial x} \coloneqq \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \frac{\partial y_2}{\partial x} \\ \vdots \\ \frac{\partial y_m}{\partial x} \end{bmatrix} \in \mathbb{R}^m, \quad \frac{\partial \mathbf{A}}{\partial x} \coloneqq \begin{bmatrix} \frac{\partial a_{11}}{\partial x} & \frac{\partial a_{12}}{\partial x} & \cdots & \frac{\partial a_{1q}}{\partial x} \\ \frac{\partial a_{21}}{\partial x} & \frac{\partial a_{22}}{\partial x} & \cdots & \frac{\partial a_{2q}}{\partial x} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_{p1}}{\partial x} & \frac{\partial a_{p2}}{\partial x} & \cdots & \frac{\partial a_{pq}}{\partial x} \end{bmatrix} \in \mathbb{R}^{p\times q} \tag{1.6}$$

である．次にベクトルで偏微分する場合,

$$\frac{\partial y}{\partial \mathbf{x}} \coloneqq \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{1\times n}, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \coloneqq \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m\times n} \tag{1.7}$$

である．ここで $\nabla_{\mathbf{x}} y(\mathbf{x}) \coloneqq \left(\frac{\partial y}{\partial \mathbf{x}}\right)^{\top}$ は y に対する $\mathbf{x}$ の勾配 (gradient) と呼ばれる．また，$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ は $\mathbf{y}$ に対する $\mathbf{x}$ の Jacobian 行列と呼ばれる．最後に，行列で偏微分する場合，

$$\frac{\partial y}{\partial \mathbf{A}} = \begin{bmatrix} \frac{\partial y}{\partial a_{11}} & \frac{\partial y}{\partial a_{21}} & \cdots & \frac{\partial y}{\partial a_{p1}} \\ \frac{\partial y}{\partial a_{12}} & \frac{\partial y}{\partial a_{22}} & \cdots & \frac{\partial y}{\partial a_{p2}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial a_{1q}} & \frac{\partial y}{\partial a_{2q}} & \cdots & \frac{\partial y}{\partial a_{pq}} \end{bmatrix} \in \mathbb{R}^{q \times p} \tag{1.8}$$

である．これは $\nabla_{\mathbf{A}} y(\mathbf{A}) \coloneqq (\frac{\partial y}{\partial \mathbf{A}})^{\top}$ とも表記する．

1.3.3 微分方程式

微分方程式はある関数とそれを微分した導関数の関係式であり，関数の特定の変数に対する変化を記述することができる．まず，1 階線形微分方程式を例として見てみよう．

$$\frac{dx(t)}{dt} = a_c x(t) + b_c u(t) \tag{1.9}$$

状態変数 $x(t)$ は，時間 t に対する関数である．添え字の c は連続 (continuous) を意味するが，これは後で離散化する際に区別するためである．この方程式においては $b_c = 0$ の場合を**同次方程式**,$b_c \neq 0$ の場合を**非同次方程式**という．

微分方程式の解

微分方程式を解くとは $x(t)$ のような関数の具体的な式を求めることである．上式の解は

$$x(t) = e^{a_c t} x(0) + \int_0^t e^{a_c (t-\tau)} b_c u(\tau) d\tau \tag{1.10}$$

として与えられる．微分方程式を解く手法は様々で，それぞれの方程式について適切な手法を選択する．本書では Laplace 変換を多用するが，細かい説明は付録にて行う．

連立線形微分方程式

n 個の微分方程式連立線形微分方程式という．これをベクトル，行列を用いて時不変 (time-invariant) の定数行列を $\mathbf{A}_c \in \mathbb{R}^{n \times n}, \mathbf{B}_c \in \mathbb{R}^{n \times m}$, 状態ベクトルを $\mathbf{x}(t) \in \mathbb{R}^n$, 入力ベクトルを $\mathbf{u}(t) \in \mathbb{R}^m$ とする．

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c \mathbf{u}(t) \tag{1.11}$$

解は

$$\mathbf{x}(t) = e^{t\mathbf{A}_c} \mathbf{x}(0) + \int_0^t e^{(t-\tau)\mathbf{A}_c} \mathbf{B}_c \mathbf{u}(\tau) d\tau \tag{1.12}$$

Laplace 変換

Laplace 変換は、与えられた時間領域の関数 $f(t)$ を複素数変数 s の関数 $F(s)$ に写像する積分変換である。特に、線形微分方程式の解析や制御工学において非常に有効な手法であり、Fourier 変換と密接な関係をもつ。Laplace 変換は、実時間領域 $t \geq 0$ 上で定義された関数 $f(t)$ に対して、以下のように定義される：

$$F(s) \coloneqq \mathscr{L}(f(t)) = \int_0^\infty f(t)\, e^{-st} dt \tag{1.13}$$

ここで $s \in \mathbb{C}$ は複素数変数であり、通常は $s = \sigma + i\omega$ の形をとる。変換核 e^{-st} を掛けて積分することにより、関数 $f(t)$ の無限大での振る舞いを抑制し、積分を収束させる効果を持つ。特に、$f(t)$ が指数関数的増加を含む場合でも、e^{-st} による減衰によってその成分を抑えることが可能となる。Laplace 変換の大きな利点の一つは、**微分演算を代数演算に変換できる**という性質にある。すなわち、$f(t)$ の微分 $\frac{d}{dt}f(t)$ に対する Laplace 変換は、次のように与えられる：

$$\mathscr{L}\left(\frac{df}{dt}\right) = sF(s) - f(0) \tag{1.14}$$

この性質により、常微分方程式は Laplace 変換の下で代数方程式に変換され、解の導出が容易となる。初期値を含んだ微分方程式を直接的に解くことができるため、初期値問題への応用にも適している。さらに、Laplace 変換には線形性：

$$\mathscr{L}(af(t) + bg(t)) = a\mathscr{L}(f(t)) + b\mathscr{L}(g(t)) \tag{1.15}$$

および畳み込みに関する定理：

$$\mathscr{L}(f * g)(t) = F(s)G(s) \tag{1.16}$$

など、多くの有用な性質がある。これにより、時間領域での複雑な演算が周波数領域で簡単な演算として扱えるようになる.

1 階線形行列微分方程式の解

時不変 (time-invariant) の定数行列を $\mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{B} \in \mathbb{R}^{n\times m}$, 状態ベクトルを $\mathbf{x}(t) \in \mathbb{R}^n$, 入力ベクトルを $\mathbf{u}(t) \in \mathbb{R}^m$ とする.

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{1.17}$$

この線形行列微分方程式を Laplace 変換 $\mathscr{L}$ を用いて解こう. $\boldsymbol{X}(s) \coloneqq \mathscr{L}(\mathbf{x}(t)), \boldsymbol{U}(s) \coloneqq \mathscr{L}(\mathbf{u}(t))$ とすると,

$$s\boldsymbol{X}(s) - \mathbf{x}(0) = \mathbf{A}\boldsymbol{X}(s) + \mathbf{B}\boldsymbol{U}(s) \tag{1.18}$$

$$(s\mathbf{I} - \mathbf{A})\boldsymbol{X}(s) = \mathbf{x}(0) + \mathbf{B}\boldsymbol{U}(s) \tag{1.19}$$

$$\boldsymbol{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}(\mathbf{x}(0) + \mathbf{B}\boldsymbol{U}(s)) \tag{1.20}$$

$$\tag{1.21}$$

行列指数関数 (matrix exponential) は

$$e^{\mathbf{A}} = \exp(\mathbf{A}) \coloneqq \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{A}^k = \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \cdots \tag{1.22}$$

として定義される．天下り的だが,

$$\mathscr{L}(e^{at}) = \frac{1}{s-a} \tag{1.23}$$

$$\mathscr{L}(e^{t\mathbf{A}}) = (s\mathbf{I} - \mathbf{A})^{-1} \tag{1.24}$$

$$\tag{1.25}$$

であるので．よって

$$\boldsymbol{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}(\mathbf{x}(0) + \mathbf{B}\boldsymbol{U}(s)) \tag{1.26}$$

$$= (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\boldsymbol{U}(s) \tag{1.27}$$

$$\mathbf{x}(t) = e^{t\mathbf{A}}\mathbf{x}(0) + \int_0^t e^{(t-\tau)\mathbf{A}}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{1.28}$$

となる．最後の式は両辺を逆 Laplace 変換した．ここで，$\mathscr{L}^{-1}(F(s)G(s)) = \int_0^t f(\tau)g(t-\tau)d\tau$ であることを用いた．区間 $[t, t+\Delta t]$ において入力 $\mathbf{u}(t)$ が一定であると仮定すると,

$$\mathbf{x}(t+\Delta t) = e^{(t+\Delta t)\mathbf{A}}\mathbf{x}(0) + \int_0^{t+\Delta t} e^{(t+\Delta t-\tau)\mathbf{A}}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{1.29}$$

$$= e^{\Delta t\mathbf{A}}e^{t\mathbf{A}}\mathbf{x}(0) + e^{\Delta t\mathbf{A}}\int_0^t e^{(t-\tau)\mathbf{A}}\mathbf{B}\mathbf{u}(\tau)d\tau + \int_t^{t+\Delta t} e^{(t+\Delta t-\tau)\mathbf{A}}\mathbf{B}\mathbf{u}(\tau)d\tau \tag{1.30}$$

$$\approx \underbrace{e^{\Delta t\mathbf{A}}}_{=:\mathbf{A}_d}\mathbf{x}(t) + \underbrace{\left[\int_t^{t+\Delta t} e^{(t+\Delta t-\tau)\mathbf{A}}d\tau\right]\mathbf{B}}_{=:\mathbf{B}_d}\mathbf{u}(t) \tag{1.31}$$

$$= \mathbf{A}_d\mathbf{x}(t) + \mathbf{B}_d\mathbf{u}(t) \tag{1.32}$$

$$\tag{1.33}$$

となる．添え字の d は離散化 (discretization) を意味する．$\mathbf{A}_c$ が正則行列の場合,

$$\mathbf{B}_d = \left[\int_t^{t+\Delta t} e^{(t+\Delta t-\tau)\mathbf{A}}d\tau\right]\mathbf{B} \tag{1.34}$$

$$= \mathbf{A}^{-1}\left[e^{\Delta t\mathbf{A}} - \mathbf{I}\right]\mathbf{B} \tag{1.35}$$

が成り立つ.

1.3.4 確率論

確率論の基本的な対象は**確率分布 (probability distribution)** である。確率分布は、ある確率変数がどのような値をどの程度の確率でとるかを定量的に記述するものである。確率変数 x は、離散的あるいは連続的な値をとる場合があり、それぞれに応じて確率分布の定義も異なる。離散的な場合、確率分布は**確率質量関数** (probability mass function; PMF) により定義され、任意の値 x に対して $p(x)$ はその値が観測される確率を与える。このとき、全ての確率の総和は 1 に等しくなければならない：

$$\sum_{x} p(x) = 1 \tag{1.36}$$

この代表例として**ポアソン分布 (Poisson distribution)** がある。ポアソン分布は、ある固定時間・空間内における稀な離散事象の発生回数をモデル化するものであり、以下のように定義される：

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad x = 0, 1, 2, \ldots \tag{1.37}$$

ここで $\lambda > 0$ は単位時間（または空間）あたりの平均発生回数を表す。この分布は事象が独立かつ一定の発生率で起きると仮定する場面で用いられる。一方、連続的な場合には**確率密度関数** (probability density function; PDF) を用いて定義される。確率密度関数 $p(x)$ は特定の値における確率そのものではなく、ある範囲に入る確率を積分によって与える関数である。たとえば、区間 $[a, b]$ における確率は次のように表される：

$$\mathbb{P}(a \leq x \leq b) = \int_a^b p(x)\, dx \tag{1.38}$$

確率密度関数もまた、定義域全体にわたる積分が 1 でなければならない：

$$\int p(x)\, dx = 1 \tag{1.39}$$

この典型例として**正規分布 (normal distribution)** が挙げられる。正規分布は、多くの自然現象や測定誤差の分布を記述するのに適しており、平均 μ、分散 σ^2 をパラメータとして次のように定義される：

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1.40}$$

この分布は平均値 μ を中心に左右対称であり、確率変数の分布が「中央に集まり、端にいくほど稀になる」という性質を持つ。特に $\mu = 0$,$\sigma^2 = 1$ の場合は標準正規分布と呼ばれる．また，正規分布の概念は一変数の場合に限らず、多次元の確率変数にも拡張される。これが**多変量正規分布 (multivariate normal distribution)** であり、ベクトル値の確率変数がとる値の分布を記述する。d 次元の確率変数 $\mathbf{x} \in \mathbb{R}^d$ が平均ベクトル $\boldsymbol{\mu} \in \mathbb{R}^d$、共分散行列 $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times d}$ をもつ多変量正規分布に従うとき、その確率密度関数は以下のように定義される：

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right) \tag{1.41}$$

ここで、$\mathbf{x}$ は d 次元の確率変数ベクトル、$\boldsymbol{\mu}$ は平均ベクトルであり、各成分が $\mathbf{x}$ の平均値，$\boldsymbol{\Sigma}$ は対象対称かつ正定値な共分散行列であり、成分 Σ_{ij} は $\mathrm{Cov}(x_i, x_j)$ を表す．この分布は、各変数が正規分布に従い、かつそれらの間の線形な関係（共分散）もモデル化できる点で、非常に広範に用いられる。特に共分散行列が対角行列のとき、すなわち変数間が独立な場合には、各変数は独立な一変量正規分布に従う。確率論においては、不確実性を定量的に扱うための基本的な概念がいくつか存在する。以下では、期待値、情報量、エントロピー、Kullback-Leibler 情報量、そして相互情報量について簡単に説明を行う。まず、**期待値 (Expectation)** は、確率変数 x に関する関数 $f(x)$ の平均値を、x の確率分布 $p(x)$ に基づいて計算する操作である。連続値の場合、期待値は次のように定義される。

$$\mathbb{E}_{x\sim p(x)}\left[f(x)\right] \coloneqq \int f(x)p(x)\,dx \tag{1.42}$$

ここで $x \sim p(x)$ は、x が分布 $p(x)$ に従うことを表す。文脈が明確な場合には、簡略に $\mathbb{E}_{p(x)}[f(x)]$ や $\mathbb{E}[f(x)]$ と表記する。次に、**情報量 (Information)** は、ある特定の事象 x の出現がどれほどの「驚き」や「情報」をもたらすかを定量化するものである。情報理論の創始者である Shannon (1948) によって導入された。出現確率が低い事象ほど、多くの情報を含むと考えられる。情報量は次のように定義される。

$$\mathbb{I}(x) \coloneqq \ln\left(\frac{1}{p(x)}\right) = -\ln p(x) \tag{1.43}$$

エントロピー (Entropy) は、確率変数の持つ平均的な不確実性、すなわち平均情報量を表す。離散的な場合には和を、連続的な場合には積分を用いて定義されるが、ここでは連続的な場合を考える。エントロピーは以下のように定義される。

$$\mathbb{H}(x) \coloneqq \mathbb{E}[-\ln p(x)] = -\int p(x)\ln p(x)\,dx \tag{1.44}$$

また、条件付きエントロピー $\mathbb{H}(x|y)$ は、y が与えられたときの x の不確実性を測る指標であり、次のように定義される。

$$\mathbb{H}(x|y) \coloneqq \mathbb{E}_{x,y}[-\ln p(x|y)] \tag{1.45}$$

この期待値は、$p(x, y)$ に基づいて計算される。次に、**Kullback-Leibler 情報量 (KL divergence)** は、ある確率分布 $p(x)$ と別の分布 $q(x)$ の間の「距離」あるいは「ずれ」を測る尺度である。対称性は持たないため、厳密には距離ではないが、情報理論や機械学習において極めて重要な概念である。KL ダイバージェンスは以下のように定義される。

$$\begin{aligned} D_{\mathrm{KL}}(p(x)\|q(x)) &\coloneqq \int p(x)\ln\frac{p(x)}{q(x)}dx && (1.46)\\ &= \int p(x)\ln p(x)\,dx - \int p(x)\ln q(x)\,dx && (1.47)\\ &= \mathbb{E}_{x\sim p(x)}[\ln p(x)] - \mathbb{E}_{x\sim p(x)}[\ln q(x)] && (1.48)\\ &= -\mathbb{H}(x) - \mathbb{E}_{x\sim p(x)}[\ln q(x)] && (1.49) \end{aligned}$$

最後に、**相互情報量 (Mutual Information)** は、二つの確率変数 x と y の間にどれほどの情報的関連性があるか、すなわち y を知ることによって x の不確実性がどれほど減少するかを定量化する。相互情報量は、エントロピーの差として次のように定義される。

$$\mathbb{I}(x;y) \coloneqq \mathbb{H}(x) - \mathbb{H}(x|y) \tag{1.50}$$

これはまた、対称的な形でも書ける。

$$\mathbb{I}(x;y) = \mathbb{H}(x) + \mathbb{H}(y) - \mathbb{H}(x,y) \tag{1.51}$$

あるいは、確率分布の比を使って次のようにも表現される。

$$\mathbb{I}(x;y) = \int p(x,y) \ln \frac{p(x,y)}{p(x)p(y)} dxdy \tag{1.52}$$

この表現は、相互情報量が、$p(x,y)$ と $p(x)p(y)$ の KL ダイバージェンスであることを示しており、すなわち独立であれば情報共有はゼロであることを意味する。

1.3.5 確率過程

確率過程（stochastic process）とは、時間とともに変化する確率変数の集まりを指す。日常においても、株価の変動、気温の変化、人の行動など、未来の状態が確実には予測できず、ある程度の不確実性を含む現象は数多く存在する。これらの確率的な時間変化を数学的に扱う枠組みが確率過程である。形式的には、確率過程とは、ある時間 t における確率変数 X_t の集まり $\{X_t\}_{t\in\mathcal{T}}$ のことである。ここで、$\mathcal{T}$ は時間を表す集，たとえば $\mathcal{T} = \{0,1,2,\dots\}$ や $\mathcal{T} = [0,\infty)$ であり，各 X_t はある量（たとえば位置や価格など）を表す確率変数である。**Markov 過程**（Markov process）は、確率過程の中でも特に重要なクラスに属する。最大の特徴は、「現在の状態が分かれば、未来の状態の予測に過去の情報（履歴）は不要である」という性質、すなわち **Markov 性**を持つことである。この性質は次のように定式化される：

$$P(X_{t+1} = x \mid X_t = x_t, X_{t-1} = x_{t-1}, \dots, X_0 = x_0) = P(X_{t+1} = x \mid X_t = x_t) \tag{1.53}$$

すなわち、未来の状態 X_{t+1} の確率は、現在の状態 X_t のみに依存し、それ以前の状態には依存しない。これにより、Markov 過程は過去の情報を逐一保持する必要がなく、解析やシミュレーションが容易になるという利点を持つ。**Wiener 過程** (Wiener process) は、連続時間 Markov 過程の代表例であり、ブラウン運動とも呼ばれる。微粒子が液体中で不規則に運動する現象（ブラウン運動）を数学的に記述するために導入されたものであるが、現在では金融や情報理論をはじめとした多くの分野において基本的なモデルとなっている。Wiener 過程 $\{W_t\}_{t\geq 0}$ は、以下の性質を満たす確率過程である：

- 初期値が 0 である：$W_0 = 0$。
- 増分の独立性：任意の $0 \leq t_1 < t_2 < \cdots < t_n$ に対し、各増分 $W_{t_{i+1}} - W_{t_i}$ は互いに独立である。

- 増分の分布が正規分布に従う：任意の $s < t$ に対して、増分 $W_t - W_s$ は平均 0、分散 $t - s$ の正規分布 $N(0, t - s)$ に従う。
- パスが連続である：時間 t に対する関数 W_t は、ほとんど確実に連続である。

このような性質により、Wiener 過程は時間的に連続かつ不規則に変化するランダムな運動を表現することができる。また、Wiener 過程はガウス過程でもあり、その共分散関数は $\mathbb{E}[W_s W_t] = \min(s, t)$ で与えられる。さらに、Wiener 過程はスケーリングの性質も持つ。すなわち、任意の $c > 0$ に対して、$\{\sqrt{c} W_{t/c}\}_{t \geq 0}$ もまた Wiener 過程である。このような対称性により、Wiener 過程は確率解析における中心的な対象として位置づけられている。以上のように、確率過程はランダムな時間変化を記述するための基本概念であり、Markov 過程や Wiener 過程はその中でも特に重要な例である。それぞれの性質を理解することは、確率論や統計物理、数理ファイナンス、機械学習などの応用分野において基礎的かつ不可欠である。

1.3.6 確率微分方程式

神経活動を含む生体活動には様々なゆらぎ（ノイズ）が常に存在しており、神経モデルにおいてもこれを考慮する必要がある。神経活動のダイナミクスを連続時間で記述する際には，決定論的な時間変化に加えて確率的なゆらぎ（ノイズ）を含む微分方程式，すなわち確率微分方程式（Stochastic Differential Equation; SDE）を用いることがある。SDE の一般的な形は以下のように与えられる：

$$dX(t) = f(X(t), t)\, dt + g(X(t), t)\, dW(t) \tag{1.54}$$

ここで、$f(X(t), t)$ はドリフト項と呼ばれる決定論的な変化、$g(X(t), t)$ は拡散項と呼ばれるノイズの強度を表す関数、$W(t)$ は標準ブラウン運動（Wiener 過程）である。$W(t)$ は連続時間の確率過程であり、その増分 $W(t + \Delta t) - W(t)$ は平均 0、分散 Δt の正規分布 $\mathcal{N}(0, \Delta t)$ に従う。

- 離散モデルでのノイズ $\mathbf{w}_t$,$\mathbf{v}_t$ は各離散時刻ごとにガウス分布から独立にサンプルされるもので、各離散時点に明示的に加えられる雑音です。
- 一方、連続モデルのノイズ $d\mathbf{w}(t), d\mathbf{v}(t)$ はブラウン運動の微小増分を表しており、連続時間での微小な変動をモデル化しています。
- このように連続時間モデルと離散時間モデルは形式上対応しています。離散化する際には通常、

$$\mathbf{w}_t \approx \int_t^{t+\Delta t} d\mathbf{w}(s), \quad \mathbf{v}_t \approx \int_t^{t+\Delta t} d\mathbf{v}(s) \tag{1.55}$$

という対応を用いて導出します。神経活動には**ノイズ**（neuronal noise）が常に存在しており、神経モデルにおいてもこれを考慮する必要がある。そのため、シナプス入力にノイズを加えることがある。たとえば、Leaky Integrate-and-Fire（LIF）モデルにおける膜電位の力学にノイズを加える場合を考える。ノイズ $\xi(t)$ を平均 $\tilde{\mu}$、分散 $\tilde{\sigma}^2$ の正規分布 $\mathcal{N}(\tilde{\mu}, \tilde{\sigma}^2)$ に従うガウシアンノイズとすると、膜電位 $V_m(t)$ の時

間発展は次式で記述される：

$$\tau_m \frac{dV_m(t)}{dt} = -(V_m(t) - V_{\text{rest}}) + R_m I(t) + \xi(t) \tag{1.56}$$

このように、線形のドリフト項 $-(V_m(t) - V_{\text{rest}})$ とガウシアンノイズ項 $\xi(t)$ を含む確率微分方程式（stochastic differential equation; SDE）で表される確率過程は、**Ornstein – Uhlenbeck（OU）過程**と呼ばれる。ノイズ $\xi(t)$ が標準正規分布 $\mathcal{N}(0,1)$ に従うホワイトノイズ $\eta(t)$ を用いて $\xi(t) = \tilde{\mu} + \tilde{\sigma}\eta(t)$ と表すこともできる。さらに、$\xi(t)$ が発火率 λ のポアソン過程に従う場合を考える。シナプス前細胞の数を N_{pre}、i 番目のシナプスにおけるシナプス強度に比例する定数を J_i とすると、ノイズの平均と分散はそれぞれ $\tilde{\mu} = \langle J_i \rangle N_{\text{pre}} \cdot \lambda$、$\tilde{\sigma}^2 = \langle J_i^2 \rangle N_{\text{pre}} \cdot \lambda$ と書ける。ただし、$\langle \cdot \rangle$ は平均を意味する。このような連続的なガウス過程でポアソン入力を近似する手法を**拡散近似**（diffusion approximation）と呼び、これは **Campbell の定理**に基づいて導かれる。このような確率微分方程式を数値的にシミュレーションするためには、時間離散化が必要となるが、その際には注意が必要である。たとえば、ドリフト項を省略し、ノイズ項のみを残した場合、

$$\tau_m \frac{dV_m(t)}{dt} = \xi(t) \tag{1.57}$$

となる。この式を時間ステップ Δt で Euler 法により離散化すると、

$$V_m(t + \Delta t) = V_m(t) + \frac{1}{\tau_m}\xi_1(t) \tag{1.58}$$

と書ける。ここで、時間ステップを Δt から $\Delta t/2$ に変更して同様に離散化すると、

$$V_m(t + \Delta t) = V_m(t + \Delta t/2) + \frac{1}{\tau_m}\xi_1(t) \tag{1.59}$$

$$= V_m(t) + \frac{1}{\tau_m}\left[\xi_1(t) + \xi_2(t)\right] \tag{1.60}$$

となる。ノイズ項 $\xi_1(t)$ と $\xi_2(t)$ は互いに独立と仮定すると、それぞれの標準偏差は $\tilde{\sigma}/\tau_m$ であり、その和 $\xi_1(t) + \xi_2(t)$ の分散は $2\tilde{\sigma}^2$、すなわち標準偏差は $\sqrt{2}\tilde{\sigma}/\tau_m$ となる。これは時間ステップの取り方によってノイズ項の大きさが変化することを意味しており、正確なシミュレーションのためには問題となる。したがって、時間ステップに依存しないようノイズ項をスケーリングする必要があり、そのためにはノイズに $\sqrt{\Delta t}$ を掛けることで対処できる。すなわち、離散化式は以下のように修正するのが望ましい：

$$V_m(t + \Delta t) = V_m(t) + \frac{\sqrt{\Delta t}}{\tau_m}\xi_1(t) \tag{1.61}$$

このように修正することで、時間ステップに依存しない安定なノイズスケーリングが可能となる。このように確率微分方程式を Euler 法で離散化する方法は、**Euler – Maruyama 法**と呼ばれる．他の離散化手法としては、Milstein 法なども存在する。このような SDE を解析的に解くことは一般に困難であるため、数値的な近似解法が必要となる。Euler – Maruyama 法は、その最も基本的な手法の一つであり、常微分

方程式に対する Euler 法の自然な拡張である。時間を刻み幅 Δt で離散化し、$t_n = n\Delta t$、$X_n \approx X(t_n)$ とおくと、Euler – Maruyama 法は次のように与えられる：

$$X_{n+1} = X_n + f(X_n, t_n)\,\Delta t + g(X_n, t_n)\,\Delta W_n \tag{1.62}$$

ここで $\Delta W_n = W(t_{n+1}) - W(t_n)$ は、正規分布 $\mathcal{N}(0, \Delta t)$ に従う確率変数である。実装上は、ΔW_n を標準正規分布 $\mathcal{N}(0,1)$ に従う独立な乱数 η_n を用いて $\Delta W_n = \sqrt{\Delta t} \cdot \eta_n$ と近似する。この結果、Euler – Maruyama 法は以下のように書き換えられる：

$$X_{n+1} = X_n + f(X_n, t_n)\,\Delta t + g(X_n, t_n)\,\sqrt{\Delta t} \cdot \eta_n \tag{1.63}$$

この手法は簡便でありながら、確率過程の時間発展を模倣するのに有用であり、多くのシミュレーションにおいて第一選択となる。ただし、その強収束次数は 0.5 であり、すなわち刻み幅 Δt を小さくしても誤差は $\mathcal{O}(\sqrt{\Delta t})$ でしか減少しない。そのため、より高精度な数値解が必要な場合には、Milstein 法などの高次手法の導入が検討される。Euler – Maruyama 法は、その単純さと広範な適用性から、確率微分方程式の数値解析における基本的な出発点となる手法である。

1.4 学習に関する基礎的概念

本書のテーマの 1 つとして「学習」が挙げられる．神経科学における「学習」と機械学習における「学習」はやや異なるが，ここで両者における学習を定義しておく．神経科学の学習は共通する点として，過去の経験に基づいて，将来の行動や出力を改善するためにシステムを変化させる，という点で共通している．システムのシステムのパラメータが変化する神経科学：シナプス強度（重み）が変化する。機械学習：ネットワークの重みやバイアスなどのパラメータが更新される。異なる点として，神経科学のモデルに機械学習

1.4.1 モデルと学習・予測

機械学習 (machine learning) における**モデル** (model) とは，2 つの集合 $\mathcal{X}, \mathcal{Y}$ を仮定した際に，入力 $x \in \mathcal{X}$ を出力 $y \in \mathcal{Y}$ に変換する関数 (写像)$f : x \to y$ あるいは条件付き確率分布 $p(y|x)$ を意味する．モデルは内部に媒介変数あるいはパラメータ (parameter)θ を持ち，$\mathcal{Y}$ を設定した後に $y = f(x;\theta)$ あるいは $p(y|x;\theta)$ を満たすように θ を更新する．この過程を**学習** (learning) あるいは**訓練** (training) と呼ぶ．学習後のパラメータ θ^* を用い，x が与えられた際の y の推定値 $\hat{y}$ を $\hat{y} = f(x;\theta^*)$ あるいは $p(y|x;\theta^*)$ から取得することを**予測** (prediction) と呼ぶ．推定値の取得の方法としてはサンプリング $\hat{y} \sim p(y|x;\theta^*)$ や $\hat{y} = \operatorname{argmax} p(y|x;\theta^*)$ などが考えられる．学習の際に用いられるデータを訓練データ (training data) と呼び，学習後のモデルの予測精度の評価に用いるデータを評価データ (test data) と呼ぶ．y が既知の場合は $D = \{(x, y)\}$ は教師付きデータ (y がラベルの場合はラベル付きデータ) と呼ばれ，x と y の対応関係を学習する過程を**教師あり学習** (supervised learning) と呼ぶ．y が未知の場合，$D = \{x\}$ はラベ

ルなしデータと呼ばれ，これのみでモデルを学習する過程を**教師なし学習** (unsupervised learning) と呼ぶ．この 2 つの学習の派生として，ラベルあり・なしデータを併用する**半教師あり学習** (semi-supervised learning) や, 教師なし学習の一種であり，入力データの部分集合から他の部分集合を予測する**自己教師あり学習** (self-supervised learning) などが存在する．この他の学習手法として**強化学習** (reinforcement learning) があり，第 11 章で詳しく説明を行う．強化学習では環境の中で行動するエージェントを仮定し，状態に応じて多くの報酬を得るための行動を学習することが目的である．

1.4.2 回帰と分類

1.4.3 識別モデル・生成モデル

オンライン・オフライン学習

1.5 線形回帰

線形回帰モデル（linear regression）は、与えられた説明変数（explanatory variable）$\mathbf{x}$ に基づいて、目的変数（objective variable）y を線形に予測することを目的とする。説明変数の次元が p であるとき、線形回帰モデルは次のように表される：

$$y = w_0 + w_1 x_1 + \cdots + w_p x_p + \varepsilon = w_0 + \sum_{j=1}^{p} w_j x_j + \varepsilon \tag{1.64}$$

ここで w_0 は切片（バイアス項）、$w_1, \ldots, w_p$ は各説明変数に対する重み、ε は誤差項を表す。$p = 1$ の場合を**単回帰**（*simple regression*）、$p > 1$ の場合を**重回帰**（*multiple regression*）と呼ぶ。

1.5.1 回帰モデルの行列表現

n 個の観測データからなるデータセット $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ を考える。ここで $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}]^\top \in \mathbb{R}^p$ は i 番目の説明変数ベクトル、$y^{(i)} \in \mathbb{R}$ は対応する目的変数の値である。なお、添字 (i) は観測値を表し、添字のない x_j, w_j などはモデル内の変数を指すことに注意する。このとき、モデル全体を行列の形で次のように記述できる：

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathbb{R}^n, \quad \mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{bmatrix} \in \mathbb{R}^{n \times (p+1)}, \quad \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1} \tag{1.65}$$

これにより、回帰モデルは次のように簡潔に表される：

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon} \tag{1.66}$$

ここで $\mathbf{X}$ は**計画行列**（design matrix）、$\boldsymbol{\varepsilon}$ は誤差ベクトルである。特に、$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$、すなわち各誤差成分が独立な平均 0・分散 σ^2 の正規分布に従うと仮定すれば、$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I})$ という確率モデルが得られる。

1.5.2 最小二乗法

最小二乗法（ordinary least squares, OLS）では、観測値 $\mathbf{y}$ と予測値 $\mathbf{X}\mathbf{w}$ との差（残差）を最小にするようにパラメータ $\mathbf{w}$ を推定する。残差ベクトル $\boldsymbol{\delta} = \mathbf{y} - \mathbf{X}\mathbf{w}$ に対し、目的関数 $\mathcal{L}(\mathbf{w})$ は次のように定義される：

$$\mathcal{L}(\mathbf{w}) \coloneqq \|\boldsymbol{\delta}\|^2 = \boldsymbol{\delta}^\top \boldsymbol{\delta} \tag{1.67}$$

この $\mathcal{L}(\mathbf{w})$ を最小化する $\mathbf{w}$ を求めることで、最適な重み $\hat{\mathbf{w}}$ を得る。最適解の推定は主に**正規方程式**（normal equation）あるいは**勾配法**（gradient descent）によって行うことができる．いずれの手法でも，目的関数 $\mathcal{L}(\mathbf{w})$ の $\mathbf{w}$ について微分、すなわち勾配 (gradient)$\nabla\mathcal{L}(\mathbf{w})$ が必要となる．$\nabla\mathcal{L}(\mathbf{w})$ は以下のように計算できる：

$$\begin{aligned}
\nabla\mathcal{L}(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} \left[(\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \right] && (1.68)\\
&= \frac{\partial}{\partial \mathbf{w}} \left(\mathbf{y}^\top\mathbf{y} - 2\mathbf{y}^\top\mathbf{X}\mathbf{w} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w} \right) && (1.69)\\
&= -2\mathbf{X}^\top\mathbf{y} + 2\mathbf{X}^\top\mathbf{X}\mathbf{w} && (1.70)\\
&= -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) && (1.71)
\end{aligned}$$

正規方程式による解析解

目的関数の勾配について $\nabla\mathcal{L}(\mathbf{w}) = 0$ となる解を $\hat{\mathbf{w}}$ とすると，次の**正規方程式**（normal equation）が得られる：

$$\mathbf{X}^\top\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^\top\mathbf{y} \tag{1.72}$$

この方程式を解くことで、パラメータの推定値 $\hat{\mathbf{w}}$ は次のように求まる：

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \tag{1.73}$$

なお、$A^+ \coloneqq (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ は $\mathbf{X}$ の **Moore – Penrose 擬似逆行列**（pseudoinverse）と呼ばれ、この表現を用いると $\hat{\mathbf{w}} = A^+\mathbf{y}$ と簡潔に記述できる。

勾配法による数値的推定

最小二乗法に基づくパラメータ推定は、数値的には**勾配法**（gradient descent）によっても実現できる。目的関数の勾配 $\nabla\mathcal{L}(\mathbf{w})$ を用いると、更新式は次のように与えられる：

$$\Delta\mathbf{w} \propto -\nabla\mathcal{L}(\mathbf{w}) = 2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \tag{1.74}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \cdot \frac{1}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) \tag{1.75}$$

ここで α は**学習率**（learning rate）と呼ばれるハイパーパラメータである。

1.5.3 リッジ回帰

線形回帰においては、説明変数が高次元である場合や、多重共線性（説明変数間の相関）が存在する場合などに、最小二乗法による推定が不安定になることがある。これに対処する手法として、**L2 正則化**を加えた**リッジ回帰**（ridge regression）が用いられる。リッジ回帰では、目的関数にパラメータの二乗ノルムを加えた正則化項を導入することにより、モデルの複雑さを抑制し、過学習の防止や推定の安定化を図る。具体的には、次のような正則化付き目的関数 $\mathcal{L}_\lambda(\mathbf{w})$ を最小化する：

$$\mathcal{L}_\lambda(\mathbf{w}) = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda\mathbf{w}^\top\mathbf{w}, \tag{1.76}$$

ここで $\lambda \geq 0$ は**正則化係数**（regularization parameter）であり、モデルのあてはまりと複雑さのトレードオフを制御する。なお通常、w_0（切片）には正則化を加えないことが多いため、必要に応じて $\mathbf{w}$ の対象を $[w_1, \ldots, w_p]^\top$ に限定する処理を行う。

正規方程式による解

L2 正則化付きの目的関数を $\mathbf{w}$ で微分して 0 に等しいとおくと、次のような修正された正規方程式が得られる：

$$(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\hat{\mathbf{w}} = \mathbf{X}^\top\mathbf{y}, \tag{1.77}$$

ここで $\mathbf{I} \in \mathbb{R}^{(p+1)\times(p+1)}$ は単位行列である．ただし、w_0 を正則化対象から除く場合、$\lambda\mathbf{I}$ の最初の対角成分をゼロにすることで対処する。この式を解くと、リッジ回帰におけるパラメータの推定値は次のように求まる：

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}. \tag{1.78}$$

この推定式は、$\mathbf{X}^\top\mathbf{X}$ が特異（非正則）である場合でも、$\lambda > 0$ により逆行列の存在が保証される点で、最小二乗法に比べて数値的に安定であるという利点がある。

勾配法による推定

リッジ回帰に対しても勾配法を適用できる。目的関数 $\mathcal{L}_\lambda(\mathbf{w})$ の勾配は次のように求まる：

$$\nabla \mathcal{L}_\lambda(\mathbf{w}) = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) + 2\lambda \mathbf{w}, \tag{1.79}$$

これに基づいて、更新式は以下のように与えられる：

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \cdot \left(\frac{1}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \lambda \mathbf{w} \right), \tag{1.80}$$

または α を調整することで、λ を勾配更新の一部として組み込む方法もある。いずれにしても、正則化項によって重みの更新が抑制されることで、過学習を防ぐ効果が得られる。

1.5.4 ロジスティック回帰

本節では、非線形回帰の一種である**ロジスティック回帰** (logistic regression) について取り扱う。ロジスティック回帰は、入力 $\mathbf{x} \in \mathbb{R}^p$ に対して出力 $y \in \{0, 1\}$ を予測する**確率的な分類モデル**である。出力は事後確率 $\Pr(y = 1 \mid \mathbf{x})$ を表し、その予測にはシグモイド関数（ロジスティック関数）を用いる。

モデルの定義

ロジスティック回帰では、まず説明変数の線形結合を求める：

$$z = w_0 + \sum_{j=1}^{p} w_j x_j = \mathbf{w}^\top \mathbf{x}' \tag{1.81}$$

ここで $\mathbf{x}' \coloneqq [1, x_1, x_2, \ldots, x_p]^\top \in \mathbb{R}^{p+1}$ はバイアス項を含んだ拡張入力ベクトル、$\mathbf{w} \in \mathbb{R}^{p+1}$ はパラメータベクトルである。この線形出力 z に対して、シグモイド関数 $\sigma(z)$ を適用することで、出力の確率的解釈が得られる：

$$\Pr(y = 1 \mid \mathbf{x}) = \sigma(z) = \frac{1}{1 + \exp(-z)} \tag{1.82}$$

したがって、クラスラベル $y \in \{0, 1\}$ の**確率モデル**は次のように表される：

$$p(y \mid \mathbf{x}; \mathbf{w}) = \sigma(z)^y (1 - \sigma(z))^{1-y} \tag{1.83}$$

パラメータの推定：最尤推定

ロジスティック回帰のパラメータは**最尤推定**により求める。データ集合 $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$ に対して、対数尤度関数は以下のように定義される：

$$\ell(\mathbf{w}) = \sum_{i=1}^{n} \left[y^{(i)} \log \sigma(z^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})) \right] \tag{1.84}$$

ここで $z^{(i)} = \mathbf{w}^\top \mathbf{x}^{(i)}$ である。この尤度を最大化することで $\mathbf{w}$ を学習する。一般には閉形式解を持たないため、**勾配降下法**などの最適化手法を用いて数値的に解く。勾配は以下のように計算される：

$$\nabla \ell(\mathbf{w}) = \sum_{i=1}^{n} (y^{(i)} - \sigma(z^{(i)}))\mathbf{x}^{(i)} \tag{1.85}$$

Cox, D. R. (1958). "The regression analysis of binary sequences." Journal of the Royal Statistical Society: Series B (Methodological), 20(2), 215 – 242.

第 2 章

発火率モデルと局所学習則

第 3 章

エネルギーベースモデル

第 4 章

ニューラルネットワークと貢献度分配問題

第 5 章

再帰型ニューラルネットワークと経時的貢献度分配問題

第 6 章

神経細胞とシナプスの生物物理学的モデル

第 7 章

スパイキングニューラルネットワークの学習則

第 8 章

リザバーコンピューティング

第 9 章

神経回路網による不確実性の表現とベイズ推論

第 10 章

運動学習と最適制御

第 11 章

強化学習

第 12 章

神経系の発生と発達の数理モデル