目次

第 1 章

はじめに

1.1 本書の目的と構成

1.1.1 神経科学におけるモデルの意義

本書では, 神経科学における数理モデルの構築と Julia 言語での実装を中心的な主題とし, その背景にある計算論的理論と実践的手法を扱う. 初めに, 神経科学におけるモデルの意義について整理しておこう.

神経科学の重要な目標の一つは, ミクロからマクロまで, また基礎から臨床にわたる幅広い研究領域とスケールの観点から, 脳神経系の構造 (structure) と機能 (function) について科学的説明を与えることである.

この目標に向けて, 自然科学に共通するように, 神経科学もまた実験と理論という二つの柱に支えられて発展してきた. 実験は観察や計測を通じて実データを得る行為であり, 理論はデータを整理・統合し, 予測や検証すべき問い (仮説) を導き出すための枠組みとなる.

モデルは？

特にモデルは, 次のような多面的な役割を担っている (**Blohm2020-vc**; **levenstein2023role**; **van2024critical**)：(1) 仮説の駆動と明示化, (2) 複雑な知見の整理と統合, (3) 観察結果の再現や予測, 仮説の提供 (4) 仮想実験の実行, (5) 科学的コミュニケーションの明確化, (6) 臨床や技術への応用可能性の提供である. このように, モデルは単なる計算装置ではなく, 科学的思考そのものの外化であり, 神経科学の進展に不可欠な知的道具である.

次に「科学的説明」ということに関してどのような観点があるのかを述べる. 神経科学における科学的説明の分類として有名なのが Marr の 3 つのレベル (Marr's Three Levels of analysis) (**Marr1982-wk**) である.

解析や理論のレベル.

Marr は〜

次に，Dayan と Abbott は〜

これらの科学的説明に関してまとめたものが次の表である.

スケール	主要な説明の観点	(**Marr1982-wk**)	(**pylyshyn1984computation**)	(**anderson1990adaptive**)	(**dayan2005theoretical**)
マクロ	なぜ (Why)	計算理論 (Computational theory)	意味論的 (Semantic)	合理的 (Rational)	解釈的 (Interpretive)
	どのように (How)	表現とアルゴリズム (Representation and algorithm)	アルゴリズム (Algorithm)	アルゴリズム (Algorithm)	記述的 (Descriptive)
			機能的構造 (Functional architecture)	実装 (Implementation)	
ミクロ	何が/何を (What)	ハードウェア実装 (Hardware implementation)	生物学的 (Biological)	生物学的 (Biological)	機構的 (Mechanistic)

表 1.1 説明・解析のレベル

説明レベル	内容	例
計算理論		
表現とアルゴリズム		
ハードウェア実装		

表 1.2 (**Marr1982-wk**) における説明・解析のレベル

説明レベル	内容	例
解釈的		
記述的		
機構的		

表 1.3 (**dayan2005theoretical**) における説明・解析のレベル

分析レベル

とはいえ，あるモデルは何に属するか，というのは明確に区別づけられないため，本書では「これは何モデル」と明示的に記載はしない.

本書では，これらの 3 つのモデルの

合理的解析 (Rational analysis) では，システムが環境下で特定の目的を「合理的」に達成すると仮定した際に，その行動を支配する制約条件や，計算が達成すべき規範を特定する.

本書では，こうした理解の枠組みを与えるために，Marr の 3 つのレベル (Marr's Three Levels) (**Marr1982-wk**) および Dayan と Abbott による 3 レベルを紹介する.

対応するモデル

科学的説明の分類は複数あるが,

解釈 (interpretive), 説明 (descriptive) and mechanistic approaches

Interpretive model (解釈的モデル, why)：行動や最適性の観点から機構を説明する (例：ベイズ推論,

強化学習)

Descriptive model (記述的モデル, how)：データの構造や統計的特徴をそのまま記述する (例：PSTH, tuning curves)

Mechanistic model (機構的モデル, what)：要素間の因果的な関係を定式化する (例：LIF モデル, STDP 則)

抽象的な意味的構成から，生体現象に即した具体的なモデル

表を参考に，統合した階層を書く.

Dayan らの序文

Theoretical analysis and computational modeling are important tools for characterizing what nervous systems do, determining how they function, and understanding why they operate in particular ways. Neuroscience encompasses approaches ranging from molecular and cellular studies to human psychophysics and psychology. Theoretical neuroscience encourages crosstalk among these subdisciplines by constructing compact representations of what has been learned, building bridges between different levels of description, and identifying unifying concepts and principles. In this book, we present the basic methods used for these purposes and discuss examples in which theoretical approaches have yielded insight into nervous system function. The questions what, how, and why are addressed by descriptive, mechanistic, and interpretive models, each of which we discuss in the following chapters.

Descriptive models summarize large amounts of experimental data compactly yet accurately, thereby characterizing what neurons and neural circuits do. These models may be based loosely on biophysical, anatomical, and physiological findings, but their primary purpose is to describe phenomena, not to explain them.

Mechanistic models, on the other hand, address the question of how nervous systems operate on the basis of known anatomy, physiology, and circuitry. Such models often form a bridge between descriptive models couched at different levels.

Interpretive models use computational and information-theoretic principles to explore the behavioral and cognitive significance of various aspects of nervous system function, addressing the question of why nervous systems operate as they do.

情報処理タスクを実行するあらゆる機械を理解するために必要な 3 つのレベル

. これは, 脳における情報処理過程を以下の三段階で整理する視点である：

計算理論 (computational theory)：対象とする現象において, 何をどのように計算するのか, 入力と出力の対応関係を定義する.

表現とアルゴリズム (representation and algorithm)：計算理論で定められた変換を, どのような内部表現と手続きにより実現するかを明示する.

実装 (implementation)：それらの手続きを, 神経回路やハードウェアといった物理基盤上でどのよう

に具現化するかを示す.

本書の立場は, これら三つのレベルを分断されたものではなく, 相互に往還可能な理解のスケールと見なすことにある. とりわけ第 (3) の実装レベルに重きを置き, モデルを読者自身の手で計算機上に構築・実行し, 理論的仮説を数値的に再現・検証する力を養うことを目標とする. その意味で本書は, 数式をコードに変換するための「実践的な翻訳辞典」としても機能する.

また, 本書で取り上げるモデルの多くは機械学習と関係している. これは神経科学と機械学習が長年にわたり相互作用してきた歴史を反映している. 神経科学から機械学習への影響には, ニューラルネットワークの構造や記憶・注意といった機能的モデルがあり, 逆に機械学習の発展が神経科学にもたらしたものとして, 強化学習に基づく意思決定理論や, ベイズ的知覚理論 (いわゆる「ベイズ脳仮説」) などが挙げられる (**Hassabis2017-zm**).

筆者の立場は, 神経科学は機械学習の素材や工学的応用のためにあるのではなく, 脳そのものの理解という自律した目的を持つ学問であるというものである. そのため本書では,「機械学習から神経科学への応用」という観点, すなわちアルゴリズム的知見を手がかりに神経過程を理解するという方向性を重視する. この視点は Blohm らが述べるように, 現象に即して問いを明確化し, モデルの仮定と評価基準を明示的に定めるという「設計としての建設的モデル化 (modeling as design) 」の理念にも通じる.

実験とは理論とは実験と理論はどのように相互作用するか. 機械学習との関連性. モデルとは説明の 3 分類とは数理モデルを扱う上でのプログラミングの重要性

1.1.2 本書の構成と読み方

本書は, 計算論的神経科学および神経モデルの理解を深めるために, Julia 言語を用いて数理モデルを実装しながら学習を進める形式を採用している. 第 1 章では, Julia 言語の基本的な使用法に加え, 本書全体で用いる数学的記法について解説する. あわせて, 神経科学における「学習」と「予測」という枠組みのもと, 本書全体の立場を概説する.

第 2・3 章では, 発火率モデルを用いた神経回路網の構成とその学習則について段階的に説明する. 第 2 章では, 神経細胞の基本的な生理学を導入し, 発火率モデルを用いた神経活動の定式化と, Hebb 則や Oja 則などの局所学習則に基づく単純なネットワークの構築について扱う. 第 3 章では, 誤差逆伝播法 (Backpropagation) に基づく現代的なニューラルネットワークを扱い, そこで発生する貢献度分配問題 (credit assignment problem) を取り上げる. さらに再帰型ニューラルネットワーク (RNN) を導入し, 時間方向での貢献度分配 (経時的貢献度分配) の問題とその学習方法を解説する.

第 4 章と第 5 章では, スパイキングニューラルネットワーク (SNN) を取り上げる. 第 4 章では, これまでのネットワークレベルの議論から個々の神経細胞とシナプスの動態に立ち戻り, スパイク発生とシナプス伝達の生物物理学的モデルを取り扱う. 第 5 章では, SNN におけるネットワーク構築と学習について, 発火率モデルで扱った学習則や誤差伝播法との接続も含めて解説する.

第 6 章以降は, 応用的・発展的トピックを各論的に扱う. 第 6 章では, リザバーコンピューティングと

いう枠組みに基づき, 複雑な動的表現を活用する発火率モデルおよびスパイキングモデルについて紹介する. 第 7 章ではネットワーク全体のエネルギーを最小化する観点に基づくエネルギーベースモデル (例えば Hopfield ネットワークやボルツマンマシン) について解説する. 神経回路網がベイズ推論を実現する可能性について, 確率的計算の関係から考察する. 第 8 章では, 運動学習における最適制御問題に対して, 脳がどのような計算を行っているかをモデルベースで探る. 第 9 章では, 強化学習の基本的枠組みと, 大脳基底核との関係について説明する.

本書は数式を交えた理論の説明ののち，コードの説明を行うという流れに基づいて執筆を行った. この通り読むことも想定しているが，初学者にはどちらかというと，コードおよび実行した後の図を先に見て,「この章を読むことでどんな結果が得られるのか」というのを先に確認したのちに数式を確認するという流れもお勧めしたい. これは，作業の目標を理解するという意味で重要なことであるからだ. 難しいと感じたら読み飛ばしてもよいし，何も理解しなくてもコードを写経するだけでも良いだろう.

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/) にアクセスし，‘install‘

現在は https://julialang.org/install/ で Juliaup を使用することが推奨されている．個別に https://julialang.org/downloads/ から使用している OS に応じて manual download を行う．

て使用している OS の download で

juliaup (https://github.com/JuliaLang/juliaup) でバージョン管理可能である．`juliaup update`

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

1.3 Julia 言語の基本構文

1.3.1 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．

1.3.2 変数名

変数名	説明
`nt`	時間ステップ数 (number of time steps)
`t`, `tt`	時間ステップのインデックス

1.4 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．

1.4.1 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 $\mathbb{R}$, 自然数全体を $\mathbb{N}$, 複素数全体を $\mathbb{C}$ と表す.
- スカラーは小文字・斜体 (例：x) , ベクトルは小文字・立体・太字 (例：$\mathbf{x}$) で表し, 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 (例：$\mathbf{X}$) で表す.
- 実 $n \times 1$ ベクトルの集合を $\mathbb{R}^n$, 実 $n \times m$ 行列の集合を $\mathbb{R}^{n \times m}$ と表す.
- 行列の (i, j) 成分は x_{ij} または $(\mathbf{X})_{ij}$ と表す.
- 行列の転置は $\mathbf{X}^\top$ と書く. ベクトルの要素は $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\top$ のように記す. このため，ベクトルの内積は $\mathbf{x}^\top\mathbf{y}$ で表す.
- 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 $\mathbf{0}$, 全要素が 1 のベクトルは $\mathbf{1}$ とする.
- 変数 x の関数は $f(x)$ などと書く. 変数 x とパラメータ a を特に区別する場合は $f(x; a)$ のようにセミコロン (;) で区切る. 引数を具体的に指定せず関数のみを示す場合は $f(\cdot)$ と書く.
- 定義には $:=$ を用いる (例：$f(x) := 2x$) . 右辺を定義する場合は $=:$ を用いる.
- スカラー x の絶対値は $|x|$ と表す. 行列 $\mathbf{X}$ の行列式は $|\mathbf{X}|$ と表す.

- ベクトル $\mathbf{x}$ のユークリッドノルムは $\|\mathbf{x}\| \coloneqq \sqrt{\sum_i x_i^2}$, 一般の p-ノルムは $\|\mathbf{x}\|_p \coloneqq (\sum_i |x_i|^p)^{1/p}$ と表す. 行列 $\mathbf{X}$ について行列ノルム（フロベニウスノルム）は $\|\mathbf{X}\|_F \coloneqq \sqrt{\sum_{i,j} x_{ij}^2}$ と表す.
- ベクトル・行列の微分は分子レイアウト記法 (numerator layout notation) を用いる.
- 近似は $\approx$ で表す (例：$a \approx b$) .
- 比例関係は $\propto$ を用いる (例：$a \propto b$) .
- 最大化・最小化の変数を明示する場合は $\arg\max_{x \in \mathcal{X}} f(x)$, $\arg\min_{x \in \mathcal{X}} f(x)$ のように表す.
- 平均 μ, 分散 σ^2 の 1 次元正規分布は $\mathcal{N}(\mu, \sigma^2)$ と表記し, 確率密度関数としては $\mathcal{N}(x \mid \mu, \sigma^2)$ と表す. 他の確率分布に関しては使用する際に定義を示す.
- 自然対数の底は e とし, 指数関数を $e^x = \exp(x)$, 自然対数を $\ln(x)$ と表す.
- 計算結果に影響しない定数は const. と表す.