2024 年度 修士論文

機械学習を用いた
イジング模型の相転移検出

2024 年 2 月 2 日

指導教員　藤原高徳

茨城大学大学院
理工学研究科　量子線科学専攻

学籍番号　22NM021S
氏名　須賀 勇貴

要旨

研究の要旨を書く.

目次

第 1 章

はじめに

1.1 研究背景

近年、急速なデジタル技術の進展と共に AI（人工知能）が様々な分野で注目を集め，その応用範囲が急拡大している．AI 技術の進歩は，新たな解決策や知見の発見を促進し，これまで解決が難しかった課題に対するアプローチを変革している．この AI ブームは，科学や技術のみならず，物理学の分野においても大きな影響を与えている．

特に物理学の領域では，機械学習やディープラーニングなどの AI 技術が新しい研究手法として取り入れられ，研究者たちによって精力的に活用されている．これらの技術は，複雑なデータパターンや相関を検出し，理論的な予測や実験結果の解釈において有益なツールとなっている．物理学者たちは，従来の手法では到達が難しかった理論の深層を探求し，新たな物理的洞察を獲得するために機械学習を駆使している．

本論文では，機械学習の進展を背景に，物理学の中でも特にイジング模型の相転移に焦点を当て，その検出において機械学習の応用可能性を考察する．イジング模型は物理学において相転移の理解や解明において基本的な役割を果たしており，従来の手法では難解であった相転移の特定や解析において，機械学習が新たな可能性を提示している．本研究では，機械学習を駆使してイジング模型の相転移を検出し，その結果から得られた洞察に基づいて，物理学における新たな展望を提案することを目的としている．

1.2 研究目的

イジング模型は物理学において基本的かつ磁性に関して極めて重要な模型であり，その相転移現象の解明は物質の特性理解において不可欠である．本研究の主な目的は，機械学習を用いてイジング模型の相転移を検出し，これによって得られる新たな物理学的洞察を通じて，以下の点に焦点を当てる．

1. イジング模型の相転移を効果的に検出することで，これまで謎に包まれていた様々な相転移現象について理解が深まる．従来の手法では難解であった相転移の特定や解析が，機械学習

の高度なパターン認識能力によって進展し，新たな物理学的洞察が得られることが期待される．

2. 機械学習の適用により，イジング模型の相転移に関連する特性やパラメータの明確な理解が可能となる．これによって，物理学者は実験結果をより効果的に解釈し，物質の複雑な相転移に関する理論的なモデルの精緻化を進めることができる．
3. 機械学習によって得られた相転移の検出結果をもとに，新たな物理学的洞察を提供することで，物理学の発展に寄与する．これにより，先駆的な研究の可能性が拡大し，新しい物理学的原則や応用が発見されることが期待される．

総じて，本研究は機械学習を通じてイジング模型の相転移を検出することで，物理学の分野における基本的な理解を深め，未知の相転移現象に対する洞察を提供することを目指している．

第 2 章

イジング模型の相転移

2.1 相転移，臨界現象とは

2.1.1 物質の三態と相転移

私たちにとって身近な物質である水 (H_2O) は，固体 (氷)，液体 (水)，気体 (水蒸気) の 3 つの状態をとる．これら状態は温度の違いによって明確な規則性があり，1 気圧の環境であれば，氷を徐々に温めていくと，ちょうど 0 ℃で氷から水へ相転移がおき，さらにその水を温めておけば，ちょうど 100 ℃で水から水蒸気への相転移が起きる．特に 1 気圧の環境では，1 気圧の環境では，水と氷が共存しうるのはちょうど 0 ℃に限られ，水と水蒸気が共存するのはちょうど 100 ℃に限られる．これが摂氏という温度目盛りの基準になっていることは言うまでもない．

液体の水の温度が 0 ℃から 100 ℃に変化する間，密度や粘性などの水の物性は少しずつ変化する．しかし，これはあくまでも定量的な変化であり，水から氷，水から水蒸気への変化のような定性的な (あるいは質的な) 変化とは違う．氷と水蒸気のそれぞれの範囲内での変化も，やはり質的な変化を伴わない，定量的な変化である．このように，定性的な変化を伴わずに移り変われるような一連の状態をひとまとめにして，相 (Phase) と呼ぶ．氷，水，水蒸気はそれぞれ固相，液相，気相という三つの相に対応する．温度などのパラメータを変化させたときに，物質が異なった相の間を移り変わる現象を相転移 (phase transition) である．

なぜ同じ H_2O という物質が，温度を変えただけで，固相，液相，気相という全く性質の異なる状態をとるのか？．素朴に考えれば，このような変化は「部品」である．H_2O 分子の性質の変化からくると考えたくなる．つまり，何らかの意味での「ミクロなルール」が変化することを想定するということである．

しかし，相転移が起こるのは「ミクロなルール」が変化するからではない．「ミクロなルール」が不変であっても，刑を構成する要素 (今回の場合なら分子) の数がきわめて大きければ，それら相互の関連が変化することで，マクロな性質の不連続な変化が生じうる．つまり，相転移は，無数の要素が複雑に絡み合ったときに全体として生じる協力現象の一種なのである．

2.2　強磁性イジング模型

この説では，強磁性体の相転移を調べるために，強磁性イジング模型を定義する．イジング模型は，実在の強磁性体のモデルとしては全く忠実ではないが，教師整体での相転移の本質をつかむためには，きわめてすぐれたモデルである．つまり，相転移や臨界現象を引き起こすために必要最低限の要素だけを持ったモデルと言える．

「イジング模型」という呼び名は，このモデルの一次元でのふるまいを 1924 年の学位論文で調べたイジングにちなんだものである．ただし，モデルの発案者は当時イジングの指導教員であったレンツである．

2.2.1　モデルの定義

一辺が L の d 次元立方格子を考える．全格子点の数を $N = L^d$ とし，それぞれの格子点に $i = 1, 2, \ldots, N$ と番号をつけておく．各格子点には，上向きと下向きの二つの状態をとるスピンがのっている．格子点 i のスピンを表すスピン変数を $\sigma_i = \pm 1$ とする．$+1$ が上向きスピン，-1 が下向きスピンのに対応する．また，スピン変数 σ_i に対応する物理量を $\hat{\sigma}_i$ と書く．

系のエネルギー固有状態は，すべての格子点のスピン変数 $\{\boldsymbol{\sigma}\} = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ と列挙することで指定できる，このようなスピンの並びのことをスピン配位と呼ぶ．それぞれのスピンが二通りの状態をとるため，系の全状態数，あるいは，スピン配位の総数は 2^N である．スピン配位 $\{\boldsymbol{\sigma}\}$ に対応するエネルギー固有値を

$$E(\{\boldsymbol{\sigma}\}) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - \mu_0 H \sum_{i=1}^{N} \sigma_i \tag{2.1}$$

とする．第一項はスピン間の相互作用を表す項で，第二項は外部磁場 H とスピン磁気モーメント μ_0 の相互作用を表す項である．第一項の和 $\sum_{\langle i,j \rangle}$ は互いに隣り合う格子点 i, j すべてについての和という意味である．

ここでは相互作用定数 J の値は正とする．したがって，隣り合う格子点の組 i, j に関わる相互作用を取り出すと，

$$-J\sigma_i\sigma_j = \begin{cases} -J, & \sigma_i = \sigma_j \text{のとき} \\ J, & \sigma_i \neq \sigma_j \text{のとき} \end{cases} \tag{2.2}$$

となる．つまり，隣り合う格子点のスピンが揃う方がエネルギーが小さくなる．このように互いにスピンを揃えようとする相互作用を，強磁性的相互作用という．

2.2.2　平衡状態での物理量

ここで，イジング模型の逆温度 β での平衡状態を調べる．この場合，カノニカル分布で平衡状態を記述するのが自然である．

分配関数は

$$Z_L(\beta, H) = \sum_{(\sigma_1,\ldots,\sigma_N)} \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \tag{2.3}$$

である．和は 2^N 通りのすべてのスピン配位についてとる．後の便利のために格子サイズを L とした．物理量 $\hat{g}$ の期待値は

$$\langle \hat{g} \rangle^{\mathrm{can}}_{\beta,H} := \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1,\ldots,\sigma_N)} g_{(\sigma_1,\ldots,\sigma_N)} \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \tag{2.4}$$

である．ここで，$g_{(\sigma_1,\ldots,\sigma_N)}$ は状態 $(\sigma_1, \ldots, \sigma_N)$ における $\hat{g}$ の値である．スピン一つあたりの自由エネルギーを

$$f_L(\beta, H) := -\frac{1}{\beta H} \ln Z_L(\beta, H) \tag{2.5}$$

，物理量としての磁化を

$$\hat{m} := \frac{1}{N} \sum_{j=1}^{N} \mu_0 \hat{\sigma}_j \tag{2.6}$$

と定義する．このとき，磁化の期待値は

$$\begin{aligned}
m_L(\beta, H) &:= \langle \hat{m} \rangle^{\mathrm{can}}_{\beta,H} \\
&= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1,\ldots,\sigma_N)} m_{(\sigma_1,\ldots,\sigma_N)} \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \\
&= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1,\ldots,\sigma_N)} \frac{1}{N} \sum_{j=1}^{N} \mu_0 \sigma_j \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \\
&= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1,\ldots,\sigma_N)} \frac{1}{\beta N} \frac{\partial}{\partial H} \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \\
&= \frac{1}{\beta N} \frac{1}{Z_L(\beta, H)} \frac{\partial}{\partial H} Z_L(\beta, H) \\
&= \frac{1}{\beta N} \frac{\partial}{\partial H} \ln Z_L(\beta, H) \\
&= -\frac{\partial}{\partial H} f_L(\beta, H)
\end{aligned} \tag{2.7}$$

と書ける．これ以降，期待値 $\langle \hat{m} \rangle^{\mathrm{can}}_{\beta,H}$ のことも，単に磁化と呼ぶ．磁化とは「系がどの程度磁石になっているか」の目安である．それに対して，「系がどの程度磁石になりやすいか」の目安になるのがゼロ磁場での磁化率は

$$\chi_L(\beta) := \left. \frac{\partial m_L(\beta, H)}{\partial H} \right|_{H=0} \tag{2.8}$$

である．

2.2.3　絶対零度

まずは，絶対零度での系のふるまいを見ていく．つまり，基底状態を求めるということである．

式 (2.2) から，一つのスピンに着目すれば，$\sigma_i = \sigma_j$ のときエネルギーが最小になる．したがって，相互作用項 $-J\sigma_i\sigma_j$ を最小化する状態は，すべてのスピンが等しくなっているときであることがわかる．つまり，すべての i について $\sigma_i = +1$ もしくは $\sigma_i = -1$ のいずれかの状態である．

一方，外部磁場とスピンの相互作用項 $-\mu_0 H\sigma_i$ を最小化する状態は，磁場の符号によって変わってくる．式 (2.1) の第二項を最小化するのは，$H > 0$ なら，すべての i について $\sigma_i = 1$ とした状態であり，$H < 0$ なら，すべての i について $\sigma_i = -1$ とした状態である．$H = 0$ のときは，値は常に 0 になるため，すべてのスピン配位で最小値を与える．

以上をまとめると，$H \geq 0$ のときはすべての i について，$\sigma_i = +1$ としたものが基底状態となり，$H \leq 0$ のときはすべての i について，$\sigma_i = -1$ としたものが基底状態となる (図 2.1).

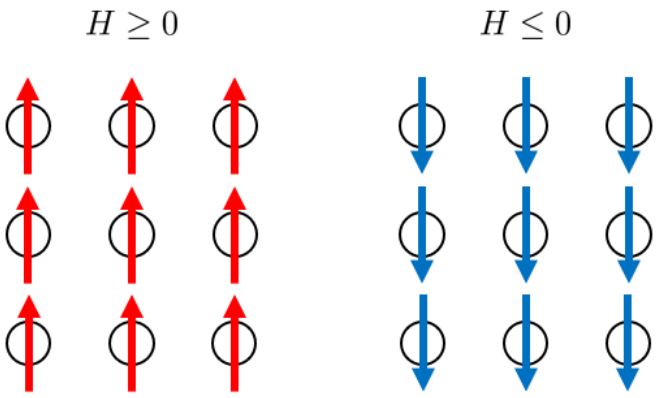


図 2.1: 絶対零度でのイジングモデルの基底状態．$H > 0$ のときはスピンはすべて上向きになり，$H < 0$ のときはスピンがすべて下向きになる $H = 0$ のときは，これら両方の状態が基底状態になる．

これに基づいて，絶対零度での磁化のふるまいを見てみる．すべてのスピンが $+1$ の状態での磁化は μ_0，すべてのスピンが -1 の状態での磁化は $-\mu_0$ なので，

$$m_L(\infty, H) = \begin{cases} -\mu_0, & H \leq 0 \text{ のとき} \\ \mu_0, & H \geq 0 \text{ のとき} \end{cases} \tag{2.9}$$

とわかる．(図 2.2) つまり，磁化は H の関数として不連続である．もちろん，これはエネルギーを最小化する状態が入れ替わったことの単純な反映にすぎない．これに対して，同じような不連続性が 0 でない温度で見られるかどうかは，本質的に難しい問題である．

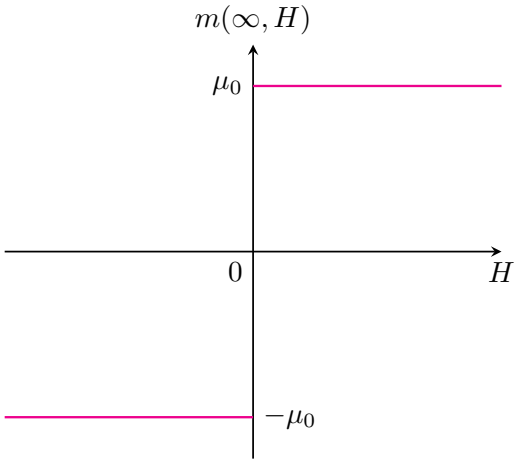


図 2.2: 絶対零度でのイジングモデルの磁化のふるまい．$H = 0$ を境に，$-\mu_0$ から μ_0 に不連続に変化する．

2.2.4 無限体積の極限

有限温度での相転移や臨界現象を調べるには，自由エネルギー $f_L(\beta, H)$ や磁化 $m_L(\beta, H)$ といった物理量を計算し，それらの振る舞いを見ればよいと思える．しかし，このとき格子サイズ L はどの程度の大きさにとればよいのか？．実は格子サイズが有限である場合，相転移を起こさないことが示せる．ここではこれを証明する．

まず，格子サイズ L が任意の有限な整数とする．このとき，式 (2.5) の形から明らかなように自由エネルギー $f_L(\beta, H)$ は β, H について何度も微分できる．よって，磁化 $m_L(\sigma, H)$ は H の連続関数であることがわかる．そして，系の対称性を考える．あるスピン配位 $(\sigma_1, \ldots, \sigma_N)$ が与えられたとき，これらすべてのスピンを反転させた新しいスピン配位を $(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_N)$ で定義する．つまり，すべての i に対して $\tilde{\sigma}_i := -\sigma_i$ とするということである．このとき，エネルギーの表式 (2.1) をスピン反転した変数を使って書き直せば，$\tilde{\sigma}_i\tilde{\sigma}_j = \sigma_i\sigma_j$ が成り立つことを使って

$$E_{(\sigma_1,\ldots,\sigma_N)} = -J\sum_{\langle i,j\rangle}\tilde{\sigma}_i\tilde{\sigma}_j - h\sum_{i=1}^{N}\tilde{\sigma}_i \tag{2.10}$$

のように，磁場を反転させた，元のエネルギーと同じ表式が得られる．つまり，すべてのスピンを反転させることは，磁場を反転させることと等価ということである．すべての $(\sigma_1, \ldots, \sigma_N)$ について足し上げることは，すべての $(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_N)$ について足し上げることと同じなので，分配関数の

表式 (2.3) を,

$$\begin{aligned} Z_L(\beta, H) &= \sum_{(\tilde{\sigma}_1,\ldots,\tilde{\sigma}_N)} \exp\left[-\beta E_{(\sigma_1,\ldots,\sigma_N)}\right] \\ &= \sum_{(\tilde{\sigma}_1,\ldots,\tilde{\sigma}_N)} \exp\left[\beta\left(J\sum_{\langle i,j\rangle}\tilde{\sigma}_i\tilde{\sigma}_j - H\sum_{i=1}^{N}\tilde{\sigma}_i\right)\right] \\ &= \sum_{(\sigma_1,\ldots,\sigma_N)} \exp\left[\beta\left(J\sum_{\langle i,j\rangle}\sigma_i\sigma_j - H\sum_{i=1}^{N}\sigma_i\right)\right] \\ &= Z_L(\beta, -H) \end{aligned} \tag{2.11}$$

となり，磁場を反転させても分配関数は変わらないという結果になる．自由エネルギーの定義である式 (2.5) より

$$f_L(\beta, H) = f_L(\beta, -H) \tag{2.12}$$

という自由エネルギーの対称性が示せる．さらに磁化は式 (2.7) より，$f_L(\beta, H)$ の H 微分で書けるので,

$$m_L(\beta, H) = -m_L(\beta, -H) \tag{2.13}$$

のように書け，磁化は磁場の反転について反対称になることがわかる．

$f_L(\beta, H)$ が微分可能なので，$m_L(\beta, H)$ はすべての β, H において定義されている．よって式 (2.13) で $H = 0$ とすれば，$m_L(\beta, 0) = -m_L(\beta, 0)$ となり，$m_L(\beta, 0) = 0$ が得られる．したがって，有限系では $0 \leq \beta < \infty$ の任意の逆温度 β で磁化がゼロになり，相転移を示さないことがわかる．

2.3　一次元イジング模型

一次元のイジング模型を考える．ハミルトニアンは

$$H = -J\sum_{i=1}^{N}\sigma_i\sigma_j - h\sum_{i=1}^{N}\sigma_i \tag{2.14}$$

なり，周期的境界条件 $\sigma_{N+1} = \sigma_1$ を課す．分配関数は

$$Z = \sum_{(\sigma_i,\ldots,\sigma_N)} \exp\left[\beta J\sum_{i=1}^{N}\sigma_i\sigma_j + \beta h\sum_{i=1}^{N}\sigma_i\right] \tag{2.15}$$

となる．ここで，次のような対称行列 T を定義する．

$$(\mathrm{T})_{\sigma_i,\sigma_{i+1}} = \exp\left[\beta J\sigma_i\sigma_{i+1} + \frac{\beta h}{2}(\sigma_i + \sigma_{i+1})\right], \tag{2.16}$$

$$\mathrm{T} = \begin{pmatrix} (\mathrm{T})_{1,1} & (\mathrm{T})_{1,-1} \\ (\mathrm{T})_{-1,1} & (\mathrm{T})_{-1,-1} \end{pmatrix} = \begin{pmatrix} e^{\beta J+\beta h} & e^{-\beta J} \\ e^{-\beta J} & e^{\beta J-\beta h} \end{pmatrix} \tag{2.17}$$

このようにすることで，分配関数 (2.15) を

$$\begin{aligned} Z &= \sum_{(\sigma_i,\ldots,\sigma_N)} (\mathrm{T})_{\sigma_1,\sigma_2}(\mathrm{T})_{\sigma_2,\sigma_3}\cdots(\mathrm{T})_{\sigma_N,\sigma_1} \\ &= \sum_{(\sigma_i,\ldots,\sigma_N)} \prod_{i=1}^{N} (\mathrm{T})_{\sigma_i,\sigma_{i+1}} \end{aligned} \tag{2.18}$$

を書くことができる．行列の積の定義より，

$$\sum_{\sigma_k=\pm 1} (\mathrm{T}^n)_{\sigma_i,\sigma_k}(\mathrm{T}^m)_{\sigma_k,\sigma_j} = (\mathrm{T}^{n+m})_{\sigma_i,\sigma_j} \tag{2.19}$$

であることを用いると，分配関数は

$$\begin{aligned} Z &= \sum_{\sigma_1=\pm 1}\sum_{\sigma_2=\pm 1}\cdots\sum_{\sigma_N=\pm 1} (\mathrm{T})_{\sigma_1,\sigma_2}(\mathrm{T})_{\sigma_2,\sigma_3}\cdots(\mathrm{T})_{\sigma_N,\sigma_1} \\ &= \sum_{\sigma_1=\pm 1}\sum_{\sigma_3=\pm 1}\cdots\sum_{\sigma_N=\pm 1} (\mathrm{T}^2)_{\sigma_1,\sigma_3}(\mathrm{T})_{\sigma_3,\sigma_4}\cdots(\mathrm{T})_{\sigma_N,\sigma_1} \\ &= \sum_{\sigma_1=\pm 1}\sum_{\sigma_4=\pm 1}\cdots\sum_{\sigma_N=\pm 1} (\mathrm{T}^3)_{\sigma_1,\sigma_4}(\mathrm{T})_{\sigma_4,\sigma_5}\cdots(\mathrm{T})_{\sigma_N,\sigma_1} \\ &= \cdots \\ &= \sum_{\sigma_1=\pm 1} (\mathrm{T}^N)_{\sigma_1,\sigma_1} \\ &= \mathrm{Tr}\left[\mathrm{T}^N\right] \end{aligned} \tag{2.20}$$

と表すことができる．対称行列 T は，スピン間の相互作用を次々と伝える役割を果たしているため，転送行列 (transfer matrix) と呼ばれる．

分配関数 (2.20) は，初等的な線形代数の知識で計算することができる．実対称であるため，転送行列 T は適当な直行行列 O を用いて，

$$\mathrm{O}^{-1}\mathrm{TO} = \begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix} \tag{2.21}$$

と対角化できる．転送行列 T の固有値は

$$\lambda_{\pm 1} = e^{\beta}\left\{\cosh\beta h \pm \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}\right\} \tag{2.22}$$

より，(2.20) はさらに

$$\begin{aligned} Z &= \mathrm{Tr}\left[\left\{\mathrm{O}\begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix}\mathrm{O}^{-1}\right\}^N\right] \\ &= \mathrm{Tr}\left[\begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix}^N\right] \\ &= (\lambda_+)^N + (\lambda_-)^N \end{aligned} \tag{2.23}$$

と表すことができる．

(2.23) より，スピン一つあたりの自由エネルギー f は

$$\begin{aligned} f &= -\frac{1}{\beta L}\ln Z \\ &= -\frac{1}{\beta L}\ln(\lambda_+)^N + (\lambda_-)^N \\ &= -\frac{1}{\beta}\ln\lambda_+ - \frac{1}{\beta L}\left\{1+\left(\frac{\lambda_-}{\lambda_+}\right)^L\right\} \end{aligned} \tag{2.24}$$

と表すことができる．ここで，$\lambda-/\lambda_+ < 1$ に注意して，$L \nearrow \infty$ とすると，

$$\begin{aligned} f &= \lim_{L\nearrow\infty} f = -\frac{1}{\beta}\ln\lambda_+ \\ &= -\frac{1}{\beta}\ln e^{\beta}\left\{\cosh\beta h \pm \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}\right\} \\ &= -1 - \frac{1}{\beta}\ln\left\{\cosh\beta h \pm \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}\right\} \end{aligned} \tag{2.25}$$

となり，無限体積極限での自由エネルギーを厳密に計算することができる．

自由エネルギーの表式 (2.25) から，磁化を計算すると，

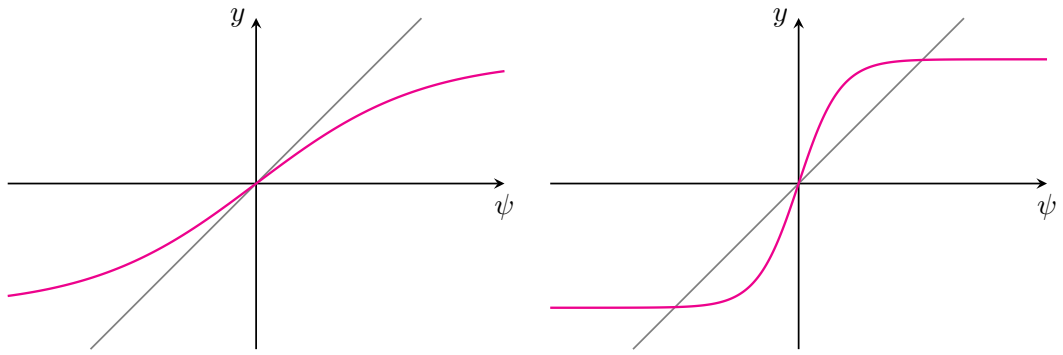
$$\begin{aligned} m(\beta,h) &= -\frac{\partial}{\partial h}f(\beta,h) \\ &= \frac{\partial}{\partial(\beta h)}\ln\left\{\cosh\beta h + \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}\right\} \\ &= \frac{1}{\cosh\beta h + \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}}\frac{\partial}{\partial(\beta h)}\left\{\cosh\beta h + \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}\right\} \\ &= \frac{1}{\cosh\beta h + \sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}}\left\{\sinh\beta h + \frac{2\sinh\beta h\cosh\beta h}{2\sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}}\right\} \\ &= \frac{\sinh\beta h}{\sqrt{(\sinh\beta h)^2 + e^{-4\beta J}}} \end{aligned} \tag{2.26}$$

となる．これは明らかに β と h について連続な関数である．特に $h = 0$ のとすれば $m(\beta, 0) = 0$ となる．つまり，有限温度の一次元イジング模型での自発磁化は 0 であり，この系は相転移を示さないことがわかる．

さらに，この結果を使って磁化率 $\chi(\beta)$ を求めると，

$$\begin{aligned}
\chi(\beta) &= \frac{\partial}{\partial h} m(\beta, h)\Big|_{h=0} \\
&= \frac{\partial}{\partial h} \frac{\sinh \beta h}{\sqrt{(\sinh \beta h)^2 + e^{-4\beta J}}}\Big|_{h=0} \\
&= \beta \frac{\partial}{\partial(\beta h)} \frac{\sinh \beta h}{\sqrt{(\sinh \beta h)^2 + e^{-4\beta J}}}\Big|_{h=0} \\
&= \beta \left[\frac{\cosh \beta h}{\sqrt{(\sinh \beta h)^2 + e^{-4\beta J}}} + \frac{(\sinh \beta h)^2 \cosh \beta h}{\{(\sinh \beta h)^2 + e^{-4\beta J}\}^{3/2}} \right]\Big|_{h=0} \\
&= \beta e^{2\beta J}
\end{aligned} \tag{2.27}$$

となる．$\beta J \ll 1$ が成り立つ高温領域では，$\chi(\beta) \simeq \beta$ となり，相互作用のない場合の振る舞い (キュリーの法則) が成り立つ．一方，低温に向かい $\beta \to \infty$ となると，相互作用のない場合の磁化率との比 $e^{2\beta J}$ は限りなく大きくなる．これは，低温で無限個のスピンが互いにそろい合おうとすることの現れとみることができる．



(a) $\beta z J \leq 1$　　**(b)** $\beta z J > 1$

図 2.3: $h = 0$ での自己整合方程式 $\psi = \tanh(\beta Z J \psi)$

2.4　二次元イジング模型

2.5　二次元イジング模型の厳密解の計算

二次元イジング模型の厳密解は Onsager によって求められた．Onsager は，転送行列を対角化することで磁場のないときの自由エネルギーを求め，比熱をある温度で発散することを示した．いくつかの解法があるが，ここでは，高温展開を用いた解法を説明する．高温展開は，温度が高いとして自由エネルギーを βJ のべきで展開する方法である．十分高温であれば，その展開を数項で打ち切って自由エネルギーの近似値とする．計算が比較的簡単なため，汎用的な手法として用いられ

ているが，近似の正当性に十分に注意する必要がある．この近似のみを用いて相転移を調べることはできない．有限項の和から特異性が生じることはないからである．本節では，無限和を計算することにより厳密解を求め，特異性が生じることを示す．

2.5.1 高温展開

$h = 0$ の場合に高温展開を行う．二次元イジング模型の分配関数は

$$Z = \mathrm{Tr} \prod_{\langle i,j \rangle} \exp\left(\beta J \sigma_i \sigma_j\right) \tag{2.28}$$

と書ける．積はスピンの最近近接についてとる．スピン変数に関する恒等式

$$\begin{aligned} e^{x\sigma_i\sigma_j} &= \frac{e^x + e^{-x}}{2} + \sigma_i\sigma_j \frac{e^x + e^{-x}}{2} \\ &= \cosh x + \sigma_i\sigma_j \sinh x \end{aligned} \tag{2.29}$$

を用いれば,

$$\begin{aligned} Z &= \mathrm{Tr} \prod_{\langle i,j \rangle} \left(\cosh \beta J + \sigma_i\sigma_j \sinh \beta J\right) \\ &= \left(\cosh \beta J\right)^{N_B} \mathrm{Tr} \prod_{\langle i,j \rangle} \left(1 + \sigma_i\sigma_j \tanh \beta J\right) \end{aligned} \tag{2.30}$$

と書ける．N_B は最近接の数を表す．$v = \tanh \beta J$ は有限温度で 1 より有限温度で 1 より小さい非負の量なので，(2.30) を v について次のように展開する．

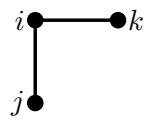
$$\prod_{\langle i,j \rangle} (1 + v\sigma_i\sigma_j) = 1 + v \sum_{\langle i,j \rangle} \sigma_i\sigma_j + v^2 \sum_{\langle i,j \rangle, \langle k,l \rangle} \sigma_i\sigma_j\sigma_k\sigma_l + \cdots \tag{2.31}$$

ここで，$\langle i,j \rangle \neq \langle k,l \rangle$ である．

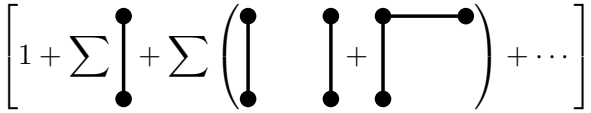
$v\sigma_i\sigma_j \longleftrightarrow$ 

(2.32)

$v^2\sigma_i\sigma_j\sigma_k\sigma_l \longleftrightarrow$ 

, $v^2\sigma_i\sigma_j^2\sigma_k \longleftrightarrow$ 

(2.33)

$Z = \left(\cosh \beta J\right)^{N_B} \mathrm{Tr}$  (2.34)

2.5.2　行列を用いた定式化

2 次元イジング模型の厳密解に向けた前段階として，行列の観点で 2 次元イジングモデルの定式化を行う．n 行 n 列の正方格子上にある $N = n^2$ 個のスピンを考える．そして，周期的境界条件を課す．

スピン座標の α 列にあるスピンをまとめて次のように表す．

$$\mu_\alpha \equiv \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \tag{2.35}$$

ここで，μ について次の境界条件が成り立つ

$$\mu_{n+1} = \mu_1 \tag{2.36}$$

また，スピン配位全体を $\{\mu_1, \ldots, \mu_n\}$ で表す．イジング模型では，α 列にあるスピンと相互作用する列は $(\alpha - 1)$ 列と $(\alpha + 1)$ 列のみある．そこで，α 列と $\alpha + 1$ 列との相互作用エネルギーを $E(\mu_\alpha, \mu_{\alpha+1})$，$\alpha$ 列のスピン内でのスピン同士の相互作用エネルギーと磁場との相互作用エネルギーの総和を $E(\mu_\alpha)$ とすると，それぞれ次のようになる．

$$E(\mu, \mu') = -\epsilon \sum_{k=1}^{n} s_k s'_k \tag{2.37}$$

$$E(\mu) = -\epsilon \sum_{k=1}^{n} s_k s_{k+1} - H \sum_{k=1}^{n} s_k \tag{2.38}$$

ここで，μ と μ' はそれぞれ，隣り合った列のスピン座標の集まりである．

$$\begin{aligned} \mu &\equiv \{s_1, \ldots, s_n\} \\ \mu' &\equiv \{s'_1, \ldots, s'_n\} \end{aligned} \tag{2.39}$$

配位 $\{\mu_1, \ldots, \mu_n\}$ における全エネルギーは

$$E_t\{\mu_1, \ldots, \mu_n\} = \sum_{\alpha=1}^{n} [E(\mu_\alpha, \mu_{\alpha+1}) + E(\mu_\alpha)] \tag{2.40}$$

となり，分配関数は

$$Z(H, T) = \sum_{\mu_1} \cdots \sum_{\mu_n} \exp\left\{ -\beta \sum_{\alpha=1}^{n} [E(\mu_\alpha, \mu_{\alpha+1}) + E(\mu_\alpha)] \right\} \tag{2.41}$$

となる．

ここで，次のように行列要素が定義された，$2^n \times 2^n$ 行列 P を考える．

$$\langle \mu | \, \mathrm{P} \, | \mu' \rangle \equiv e^{-\beta [E(\mu_\alpha, \mu_{\alpha+1}) + E(\mu_\alpha)]} \tag{2.42}$$

これを用いると分配関数は

$$Z(H,T) = \sum_{\mu_1} \cdots \sum_{\mu_n} \langle \mu_1 | \, \mathrm{P} \, | \mu_2 \rangle \langle \mu_2 | \, \mathrm{P} \, | \mu_3 \rangle \cdots \langle \mu_n | \, \mathrm{P} \, | \mu_1 \rangle \tag{2.43}$$

$$= \sum_{\mu_1} \langle \mu_1 | \, \mathrm{P}^n \, | \mu_1 \rangle = \mathrm{Tr} \ \mathrm{P}^n \tag{2.44}$$

と P^n のトレースで表せることがわかる．行列のトレースは行列の表現から独立しているため，上式のトレースは，以下のような対角化された P で評価しても結果に影響を与えない．

$$\mathrm{P} = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_{2^n} \end{bmatrix} \tag{2.45}$$

ここで，$\lambda_1, \lambda_2, \ldots, \lambda_{2^n}$ は P の固有値である．P^n もまた対角行列であり，対角成分は $(\lambda_1)^n, (\lambda_2)^n, \ldots, (\lambda_{2^n})^n$ である．したがって，分配関数は

$$Z(H,T) = \sum_{\alpha=1}^{2^n} (\lambda_\alpha)^n \tag{2.46}$$

で表される．$E(\mu, \mu')$ と $E(\mu)$ は n のオーダーであるため，式 (2.42) の形式から，n が大きい場合，P の固有値は一般に e^n のオーダーであることが予想される．$\lambda_{\max}$ を P の最大固有値とすると，

$$\lim_{n\to\infty} \frac{1}{n} \ln \lambda_{\max} = \text{finite number} \tag{2.47}$$

と予想される．そして，もしこれが正しく，すべての固有値 λ_α が正ならば，

$$(\lambda_{\max})^n \leq Z \leq 2^n (\lambda_{\max})^n \tag{2.48}$$

もしくは

$$\frac{1}{n} \ln \lambda_{\max} \leq \frac{1}{n^2} \ln Z \leq \frac{1}{n} \ln \lambda_{\max} + \frac{1}{n} \ln 2 \tag{2.49}$$

となる．したがって，はさみうちの原理より，

$$\lim_{N\to\infty} \frac{1}{N} \ln Z = \lim_{n\to\infty} \frac{1}{n} \ln \lambda_{\max} \tag{2.50}$$

か成り立つことがわかる．$N = n^2$．式 (2.47) が真であり，すべての固有値 λ_α が正であるという仮定が正しいことは後でわかる．したがって，P の最大の固有値が分かれば $n \to \infty$ での分配関数が分かる．このセクションの残りの部分は，P の明示的な表現の説明に当てられる．

2.5.3　行列 P

式 (2.42) と式 (2.39) から我々は P の行列要素を

$$\begin{aligned}\langle s_1,\ldots,s_n|\,\mathrm{P}\,|s'_1,\ldots,s'_n\rangle &= e^{-\beta[E(\mu,\mu')+E(\mu)]}\\ &= e^{\beta\sum_{k=1}^{n}(\epsilon s_k s'_k+\epsilon s_k s_{k+1}+Hs_k)}\\ &= \prod_{k=1}^{n} e^{\beta H s_k}e^{\beta\epsilon s_k s_{k+1}}e^{\beta\epsilon s_k s'_k}\end{aligned}\tag{2.51}$$

のように得た．ここで，$2^n\times 2^n$ の 3 つの行列 V'_1, V_2, V_3 を定義する．それぞれ行列要素は次のように与える．

$$\langle s_1,\ldots,s_n|\,V'_1\,|s'_1,\ldots,s'_n\rangle \equiv \prod_{k=1}^{n} e^{\beta\epsilon s_k s'_k}\tag{2.52}$$

$$\langle s_1,\ldots,s_n|\,V_2\,|s'_1,\ldots,s'_n\rangle \equiv \delta_{s_1 s'_1}\ldots\delta_{s_n s'_n}\prod_{k=1}^{n} e^{\beta\epsilon s_k s_{k+1}}\tag{2.53}$$

$$\langle s_1,\ldots,s_n|\,V_3\,|s'_1,\ldots,s'_n\rangle \equiv \delta_{s_1 s'_1}\ldots\delta_{s_n s'_n}\prod_{k=1}^{n} e^{\beta\epsilon H s_k}\tag{2.54}$$

$$\tag{2.55}$$

ここで，$\delta_{ss'}$ はクロネッカーのデルタ記号である．したがって，この表現では V_2 と V_3 は対角行列になる．それは簡単に示せる．

$$\mathrm{P} = V_3 V_2 V'_1\tag{2.56}$$

行列乗算の通常の意味で，つまり

$$\begin{aligned}&\langle s_1,\ldots,s_n|\,\mathrm{V_3V_2V'_1}\,|s'_1,\ldots,s'_n\rangle\\ &= \sum_{s''_1,\ldots,s''_n}\sum_{s'''_1,\ldots,s'''_n}\langle s_1,\ldots,s_n|\,\mathrm{V_3}\,|s''_1,\ldots,s''_n\rangle\,\langle s''_1,\ldots,s''_n|\,\mathrm{V_2}\,|s'''_1,\ldots,s'''_n\rangle\\ &\times\langle s'''_1,\ldots,s'''_n|\,\mathrm{V'_1}\,|s'_1,\ldots,s'_n\rangle\\ &= \sum_{s''_1,\ldots,s''_n}\sum_{s'''_1,\ldots,s'''_n}\delta_{s_1 s''_1}\cdots\delta_{s_n s''_n}\delta_{s''_1 s'''_1}\cdots\delta_{s''_n s'''_n}\prod_{k=1}^{n} e^{\beta H s_k}e^{\beta\epsilon s''_k s'''_{k+1}}e^{\beta\epsilon s'''_k s'_k}\\ &= \sum_{s''_1,\ldots,s''_n}\delta_{s_1 s''_1}\cdots\delta_{s_n s''_n}\prod_{k=1}^{n} e^{\beta H s_k}e^{\beta\epsilon s''_k s''_{k+1}}e^{\beta\epsilon s''_k s'_k}\\ &= \prod_{k=1}^{n} e^{\beta H s_k}e^{\beta\epsilon s_k s_{k+1}}e^{\beta\epsilon s_k s'_k}\end{aligned}$$

2.5.4　行列の直積

ここで，$\mathrm{V_3, V_2, V'_1}$ による便利な方法を述べる前に，行列の積の概念を導入する．

$m \times m$ の 2 つの行列 A, B を考える．行列要素はそれぞれ，$\langle i|\,A\,|j\rangle, \langle i|\,B\,|j\rangle$ である．i, j はそれぞれ $1, 2, \ldots, m$ の値をとる．このとき，直積 $A \times B$ は $m^2 \times m^2$ の行列となり，行列要素は

$$\langle ii'|\,A \times B\,|jj'\rangle \equiv \langle i|\,A\,|j\rangle\,\langle i'|\,B\,|j'\rangle \tag{2.57}$$

と定義する．3 つ以上の行列の直積 $A \times B \times \cdots \times C$ にも拡張でき，行列要素は

$$\langle ii' \cdots i''|\,A \times B \times \cdots \times C\,|jj' \cdots j''\rangle \equiv \langle i|\,A\,|j\rangle\,\langle i'|\,B\,|j'\rangle \cdots \langle i''|\,A\,|j''\rangle \tag{2.58}$$

で表される．もし AB が通常の行列乗算のもとでの行列 A と B の積を表すなら，次のようになる．

$$(A \times B)(C \times D) = (AC) \times (BD) \tag{2.59}$$

これは以下のように示せる．

$$\begin{aligned}\langle ii'|\,(A \times B)(C \times D)\,|jj'\rangle &= \sum_{kk'} \langle ii'|\,A \times B\,|kk'\rangle\,\langle kk'|\,C \times D\,|jj'\rangle \\ &= \langle i|\,AC\,|j\rangle\,\langle i'|\,BD\,|j'\rangle \\ &= \langle ii'|\,(AC) \times (BD)\,|jj'\rangle\end{aligned}$$

式 (**??**) を一般化した，以下の等式も成り立つ．

$$(A \times B \times \cdots \times C)(D \times E \times \cdots \times F) = (AD) \times (BE) \times \cdots \times (CF) \tag{2.60}$$

2.5.5 スピン行列

ここで，$\mathrm{V}_1', \mathrm{V}_2, \mathrm{V}_3$ を便利に表現できる特別な行列をいくつか紹介する．よく知られた 3 つの 2×2 パウリスピン行列を X, Y, Z とする．

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.61}$$

これらパウリ行列は次の関係式を満たす．

$$\begin{gathered} X^2 = Y^2 = Z^2 = 1 \\ XY + YX = 0, \;\; YZ + ZY = 0, \;\; ZX + XZ = 0 \\ XY = iZ, \;\; YZ = iX, \;\; ZX = iY \end{gathered} \tag{2.62}$$

ここで，$2^n \times 2^n$ 行列 $\mathrm{X}_\alpha, \mathrm{Y}_\alpha, \mathrm{Z}_\alpha$ を定義する．

$$\begin{aligned} \mathrm{X}_\alpha &\equiv 1 \otimes 1 \otimes \cdots \otimes \mathrm{X} \otimes \cdots \otimes 1 \\ \mathrm{Y}_\alpha &\equiv 1 \otimes 1 \otimes \cdots \otimes \mathrm{Y} \otimes \cdots \otimes 1 \\ \mathrm{Z}_\alpha &\equiv 1 \otimes 1 \otimes \cdots \otimes \mathrm{Z} \otimes \cdots \otimes 1 \end{aligned} \tag{2.63}$$

$\alpha \neq \beta$ のとき以下の関係式が成り立つ．

$$\begin{aligned} [\mathrm{X}_\alpha, \mathrm{X}_\beta] = [\mathrm{Y}_\alpha, \mathrm{Y}_\beta] = [\mathrm{Z}_\alpha, \mathrm{Z}_\beta] = 0 \\ [\mathrm{X}_\alpha, \mathrm{Y}_\beta] = [\mathrm{X}_\alpha, \mathrm{Z}_\beta] = [\mathrm{Y}_\alpha, \mathrm{Z}_\beta] = 0 \end{aligned} \tag{2.64}$$

$2^n \times 2^n$ 行列 $\mathrm{X}_\alpha, \mathrm{Y}_\alpha, \mathrm{Z}_\alpha$ は (2.62) のすべての関係式を満たす．

2.5.6　行列 $\mathrm{V}_1', \mathrm{V}_2, \mathrm{V}_3$

式 (2.52) から，V_1' が n 個の同じ 2×2 行列の直積で表せることがわかる．

$$\mathrm{V}_1' = \boldsymbol{a}\times\boldsymbol{a}\times\cdots\times\boldsymbol{a} \tag{2.65}$$

ここで,

$$\langle s|\,\boldsymbol{a}\,|s'\rangle = e^{\beta\epsilon ss'} \tag{2.66}$$

である．したがって,

$$\boldsymbol{a} = \begin{bmatrix} e^{\beta\epsilon} & e^{-\beta\epsilon} \\ e^{-\beta\epsilon} & e^{\beta\epsilon} \end{bmatrix} = e^{\beta\epsilon} + e^{-\beta\epsilon}X \tag{2.67}$$

である．ここで，α と θ を定数として

$$\boldsymbol{a} = \alpha e^{\theta X} = \alpha(\cosh\theta + X\sinh\theta) \tag{2.68}$$

として，式 (2.67) と係数比較すれば,

$$\begin{cases} e^{\beta\epsilon} = \alpha\cosh\theta \\ e^{-\beta\epsilon} = \alpha\sinh\theta \end{cases} \tag{2.69}$$

となり，これを解くことで

$$\boldsymbol{a} = \sqrt{2\sinh(2\beta\epsilon)}e^{\theta X}, \quad \tanh\theta \equiv e^{-2\beta\epsilon} \tag{2.70}$$

である．したがって

$$\mathrm{V}_1' = [2\sinh(2\beta\epsilon)]^{\frac{n}{2}} e^{\theta X}\otimes e^{\theta X}\otimes\cdots\otimes e^{\theta X} \tag{2.71}$$

と表せる．ここで,

$$e^{\theta X}\otimes e^{\theta X}\otimes\cdots\otimes e^{\theta X} \tag{2.72}$$
$$= (e^{\theta X}\otimes 1\otimes\cdots\otimes 1)(1\otimes e^{\theta X}\otimes\cdots\otimes 1)\cdots(1\otimes 1\otimes\cdots\otimes e^{\theta X}) \tag{2.73}$$
$$= e^{\theta \mathrm{X}_1}e^{\theta \mathrm{X}_2}\cdots e^{\theta \mathrm{X}_n} \tag{2.74}$$
$$= e^{\theta(\mathrm{X}_1+\mathrm{X}_2+\cdots+\mathrm{X}_n)} \tag{2.75}$$

と表せることを使う．2 つ目の等式では $1\otimes\cdots\otimes e^{\theta X}\otimes\cdots\otimes 1 = e^{\theta(1\otimes\cdots\otimes X\otimes\cdots 1)}$ を使った．これより，V_1' は次のように書ける．

$$\mathrm{V}_1' = [2\sinh(2\beta\epsilon)]^{\frac{n}{2}}\ \mathrm{V}_1 \tag{2.76}$$
$$\mathrm{V}_1 = \prod_{\alpha=1}^{n} e^{\theta \mathrm{x}_\alpha}, \quad \tanh\theta \equiv e^{-2\beta\epsilon} \tag{2.77}$$

同様な考えより，$\mathrm{V}_2, \mathrm{V}_3$ に関しても，

$$\mathrm{V}_2 = \prod_{\alpha=1}^{n} e^{\beta\epsilon \mathbf{Z}_\alpha \mathbf{Z}_{\alpha+1}} \tag{2.78}$$

$$\mathrm{V}_3 = \prod_{\alpha=1}^{n} e^{\beta H \mathbf{Z}_\alpha} \quad , \quad \mathbf{Z}_{n+1} \equiv \mathbf{Z}_1 \tag{2.79}$$

が示せる．したがって

$$\mathrm{P} = [2\sinh(2\beta\epsilon)]^{\frac{n}{2}} \mathrm{V}_3 \mathrm{V}_2 \mathrm{V}_1 \tag{2.80}$$

と表すことができる．$H = 0$ のとき，つまり $V_3 = 1$ のとき，2 次元イジング模型を完全に定式化できる．

2.5.7 数学的余談

磁場がない場合の 2 次元イジング模型の解法に関連する，一般的なクラスの行列の研究を以下に示す．

次の反交換関係を満たす $2n$ 個の行列 $\Gamma_\mu (\mu = 1, \ldots, 2n)$ を定義する．

$$\{\Gamma_\mu, \Gamma_\nu\} = \Gamma_\mu \Gamma_\nu + \Gamma_\nu \Gamma_\mu = 2\delta_{\mu\nu} \tag{2.81}$$

(a) Γ_μ の次元は $2^n \times 2^n$ より小さくならない

(b) もし，Γ_μ と Γ'_μ が 2.81 を満たす組ならば，$\Gamma_\mu = \mathrm{S}\Gamma'_\mu \mathrm{S}^{-1}$ となる正則行列 S が必ず存在する．

(c) すべての $2^n \times 2^n$ 行列は，単位行列，ガンマ行列 Γ_μ，およびガンマ行列の積 $\Gamma_\mu\Gamma_\nu, \Gamma_\mu\Gamma_\nu\Gamma_\lambda, \cdots$ の線形結合で表せる．

$n = 1$ のとき，2.81 を満たすのは 2×2 パウリスピン行列である．すべての 2×2 行列が単位行列とパウリスピン行列の線形結合で表せることは明らかである．また，$n = 2$ のときはすべての 4×4 の Dirac 行列 γ_μ となる．

$2^n \times 2^n$ 行列の可能な $\{\Gamma_\mu\}$ の表現は次のようなものである．

$$\begin{aligned}
&\Gamma_1 = \mathrm{Z}_1 && \Gamma_2 = \mathrm{Y}_1 \\
&\Gamma_3 = \mathrm{X}_1\mathrm{Z}_2 && \Gamma_4 = \mathrm{X}_1\mathrm{Y}_1 \\
&\Gamma_5 = \mathrm{X}_1\mathrm{X}_2\mathrm{Z}_2 && \Gamma_6 = \mathrm{X}_1\mathrm{X}_2\mathrm{Y}_1 \\
&\quad \vdots && \quad \vdots
\end{aligned}$$

つまり，

$$\begin{aligned}
\Gamma_{2\alpha-1} &= \mathrm{X}_1\mathrm{X}_2 \cdots \mathrm{X}_{\alpha-1}\mathrm{Z}_\alpha \quad (\alpha = 1, \ldots, n) \\
\Gamma_{2\alpha} &= \mathrm{X}_1\mathrm{X}_2 \cdots \mathrm{X}_{\alpha-1}\mathrm{Y}_\alpha \quad (\alpha = 1, \ldots, n)
\end{aligned} \tag{2.82}$$

2.5.8　厳密解

磁場がない，つまり $H = 0$ の場合，行列 P は

$$\begin{aligned} \mathrm{P} &= [2\sinh(2\beta\epsilon)]^{\frac{n}{2}}\mathrm{V}_2\mathrm{V}_1 \\ &\equiv [2\sinh(2\beta\epsilon)]^{\frac{n}{2}}\mathrm{V} \end{aligned} \tag{2.83}$$

となる．ここで，$\mathrm{V} = \mathrm{V}_1\mathrm{V}_2$ とした．式 (2.50) より，

$$\begin{aligned} \lim_{N\to\infty}\frac{1}{N}\ln Z(0,T) &= \lim_{n\to\infty}\frac{1}{n}\ln\left\{[2\sinh 2\beta\epsilon]^{\frac{n}{2}}\Lambda\right\} \\ &= \frac{1}{2}\ln[2\sinh 2\beta\epsilon] + \lim_{n\to\infty}\frac{1}{n}\ln\Lambda \end{aligned} \tag{2.84}$$

となる．ここで，Λ は $\mathrm{V} = \mathrm{V}_1\mathrm{V}_2$ の最大固有値である，V_1 は式 (**??**)，V_1 は (**??**) で与えられる．これらの式は，V の固有値がすべて正で，$\lim_{n\to\infty} n^{-1}\ln\Lambda$ が存在する場合に成り立つ．したがって行列 V を対角化を考えればよい．

2.5.9　スピン代数による V の表現

式 (**??**) より，

$$\begin{aligned} \Gamma_{2\alpha}\Gamma_{2\alpha-1} &= (\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_{\alpha-1}\mathrm{Y}_\alpha)(\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_{\alpha-1}\mathrm{Z}_\alpha) \\ &= (\mathrm{X}_1)^2(\mathrm{X}_2)^2\cdots(\mathrm{X}_{\alpha-1})^2\mathrm{Y}_\alpha\mathrm{Z}_\alpha \\ &= \mathrm{Y}_\alpha\mathrm{Z}_\alpha \\ &= i\mathrm{X}_\alpha \end{aligned} \tag{2.85}$$

$$\begin{aligned} \Gamma_{2\alpha+1}\Gamma_{2\alpha} &= (\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_\alpha\mathrm{Z}_{\alpha+1})(\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_{\alpha-1}\mathrm{Y}_\alpha) \\ &= (\mathrm{X}_1)^2(\mathrm{X}_2)^2\cdots(\mathrm{X}_{\alpha-1})^2\mathrm{X}_\alpha\mathrm{Z}_{\alpha+1}\mathrm{Y}_\alpha \\ &= \mathrm{X}_\alpha\mathrm{Z}_{\alpha+1}\mathrm{Y}_\alpha \\ &= \mathrm{X}_\alpha\mathrm{Y}_\alpha\mathrm{Z}_{\alpha+1} \\ &= i\mathrm{Z}_\alpha\mathrm{Z}_{\alpha+1} \end{aligned} \tag{2.86}$$

$$\begin{aligned} \Gamma_1\Gamma_{2n} &= \mathrm{Z}_1(\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_{\alpha-1}\mathrm{Y}_n) \\ &= -i\mathrm{Z}_1\mathrm{Z}_n(\mathrm{X}_1\mathrm{X}_2\cdots\mathrm{X}_n) \\ &\equiv -i\mathrm{Z}_1\mathrm{Z}_n\mathrm{U} \end{aligned} \tag{2.87}$$

これより

$$\mathrm{V}_1 = \prod_{\alpha=1}^{n} e^{\theta\mathrm{X}_\alpha} = \prod_{\alpha=1}^{n} e^{-i\theta\Gamma_{2\alpha}\Gamma_{2\alpha-1}} \tag{2.88}$$

$$\begin{aligned}
\mathrm{V}_2 &= \prod_{\alpha=1}^{n} e^{\beta\epsilon \mathrm{Z}_\alpha \mathrm{Z}_{\alpha+1}} \\
&= \left[\prod_{\alpha=1}^{n-1} e^{\beta\epsilon \mathrm{Z}_\alpha \mathrm{Z}_{\alpha+1}}\right] e^{\beta\epsilon \mathrm{Z}_n \mathrm{Z}_1} \\
&= e^{\beta\epsilon \mathrm{Z}_n \mathrm{Z}_1} \left[\prod_{\alpha=1}^{n-1} e^{\beta\epsilon \mathrm{Z}_\alpha \mathrm{Z}_{\alpha+1}}\right] \\
&= e^{i\beta\epsilon \mathrm{U}\Gamma_1\Gamma_{2n}} \prod_{\alpha=1}^{n-1} e^{-i\beta\epsilon\Gamma_{2\alpha+1}\Gamma_{2\alpha}}
\end{aligned} \tag{2.89}$$

したがって，$\phi = \beta\epsilon$ として

$$\mathrm{V} \equiv \mathrm{V}_1\mathrm{V}_2 = e^{i\beta\epsilon \mathrm{U}\Gamma_1\Gamma_{2n}} \left[\prod_{\alpha=1}^{n-1} e^{-i\beta\epsilon\Gamma_{2\alpha+1}\Gamma_{2\alpha}}\right] \left[\prod_{\lambda=1}^{n} e^{-i\theta\Gamma_{2\lambda}\Gamma_{2\lambda-1}}\right] \tag{2.90}$$

ここで，いくつか U の性質を書く

(a) $\mathrm{U}^2 = 1, \quad \mathrm{U}(1+\mathrm{U}) = 1+\mathrm{U}, \quad \mathrm{U}(1-\mathrm{U}) = -(1-\mathrm{U})$

(b) $\mathrm{U} = i^n\Gamma_1\Gamma_2\cdots\Gamma_{2n}$

(c) $\{\mathrm{U}, \Gamma_\mu\} = 0$

これらの性質を用いることで

$$\begin{aligned}
e^{i\phi\Gamma_1\Gamma_{2n}} &= \left[\frac{1}{2}(1+\mathrm{U}) + \frac{1}{2}(1+\mathrm{U})\right] [\cosh\phi + i\Gamma_1\Gamma_{2n}\sinh\phi] \\
&= \frac{1}{2}(1+\mathrm{U})\,[\cosh\phi + i\Gamma_1\Gamma_{2n}\sinh\phi] + \frac{1}{2}(1-\mathrm{U})\,[\cosh\phi - i\Gamma_1\Gamma_{2n}\sinh\phi] \\
&= \frac{1}{2}(1+\mathrm{U})e^{i\phi\Gamma_1\Gamma_{2n}} + \frac{1}{2}(1-\mathrm{U})e^{-i\phi\Gamma_1\Gamma_{2n}}
\end{aligned} \tag{2.91}$$

したがって

$$\mathrm{V} = \frac{1}{2}(1+\mathrm{U})\mathrm{V}^+ + \frac{1}{2}(1-\mathrm{U})\mathrm{V}^- \tag{2.92}$$

$$\mathrm{V}^\pm \equiv e^{\pm i\phi \mathrm{U}\Gamma_1\Gamma_{2n}} \left[\prod_{\alpha=1}^{n-1} e^{-i\phi\Gamma_{2\alpha+1}\Gamma_{2\alpha}}\right] \left[\prod_{\lambda=1}^{n} e^{-i\theta\Gamma_{2\lambda}\Gamma_{2\lambda-1}}\right] \tag{2.93}$$

第 3 章

機械学習と深層学習

ここでは，イジングモデルに対して機械学習，深層学習による手法を応用する上で必要となる機械学習と深層学習の基礎知識について説明する．

3.1 学習とは

機械学習 (Machine Learning) とはデータ分析の一種で，人間が行う知的作業を計算機に実行させる手法のことを意味する．機械学習では，データセットのみを機械 (コンピュータ) に与えることでデータセットの中に潜むパターンや特徴を自動で “学習” することができる．

機械 (コンピュータ) が学習を行うため，機械学習と名付けられているわけだが，“学習” とは何かをきちんと定式化する必要がある．さまざまな定式化が存在するが，ここでは，T. M. ミッチェル (Tom Michael Mitchell) の書籍 [1] で書かれている有名な定式化を紹介する．次の文章はその書籍に書かれている学習の定義の部分をそのまま日本語訳したものである．

> “コンピュータプログラムが，ある種のタスク T とパフォーマンス評価尺度 P において，経験 E から学習するとは，タスク T におけるその性能を P によって評価した際に，経験 E によってそれが改善されている場合である．” T. M. ミッチェル

ここで，タスク T は解きたい問題，パフォーマンス評価尺度 P は精度や誤差率などの評価指標のこと，経験 E はデータセットのことを指す．

この内容をもう少しかみ砕いて説明しよう．機械学習では，学習を行うための “モデル” を構築する必要がある．機械学習の分野でモデルとは，多数のパラメータを持った非線形関数

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}), \quad \boldsymbol{\theta} = \{\theta_1, \theta_2, \theta_3, \cdots\} \tag{3.1}$$

のことである．モデルのパラメータを $\boldsymbol{\theta}^* = \{\theta_1^*, \theta_2^*, \theta_3^*, \cdots\}$ と値を決めてしまえば，入力データ $\boldsymbol{x}$ を与えたときになにかしらの出力 $\boldsymbol{y} = f_{\boldsymbol{\theta}^*}(\boldsymbol{x})$ を返す．つまり，モデルの出力結果は関数の形とパラメータの値によって決まるということである．

我々は，モデルに対してデータを入力したときに，タスクをこなせるような正しい結果を出力さ

せるようにモデルのパラメータの値を適切に設定，つまりパラメータの最適化を行う必要がある．この作業を機械学習では機械 (コンピュータ) に行わせる．

まず，パラメータの初期値を乱数を用いてランダムに決め，データ入力する．当然この状態で返ってくる出力結果は本来出力してほしい正解の値からは外れた値が帰ってくる．そこで，出力と正解の誤差を測るような評価基準を決めておき，算出された評価結果を良くするようにモデルのパラメータを更新してあげることによってモデルを改善する．再びデータを入力して，評価結果からパラメータを行う．この流れを繰り返し行うことで，モデルを改善を何度も行う．そうすることで最終的には正解に近い出力結果を得られるモデルを作成する．これが学習の流れである．ここで注目してほしいのは，機械学習ではモデルの改善を機械に行わせているという点である．

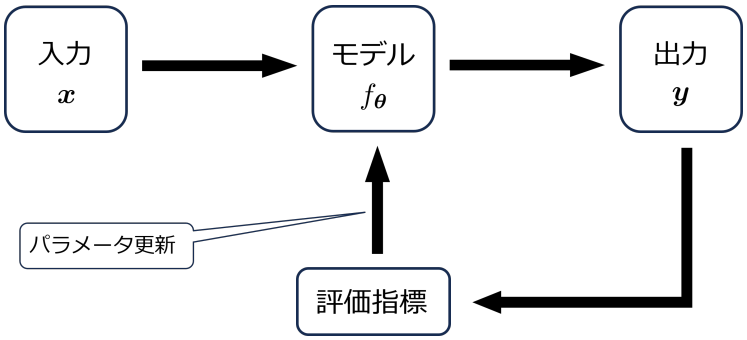


図 3.1: 機械学習の大まかな流れ

3.1.1 代表的なタスク

機械学習におけるタスク，つまり解きたい問題にはどのようなものがあるか．ここでは代表的な例である「クラス分類」と「回帰」について説明する．

クラス分類

分類 (classification) とは，データをいくつかのクラスに仕分ける作業のことである．簡単な例として，送られてきた電子メールをみて，それがスパムメールか通常メールを判別するような作業を考える．これは「スパムメール or 通常メール」という 2 つのクラスに分類するため，2 クラス分類と呼べる．スパムメールを C_0，通常メールを C_1 とラベル付けし，入力データを $\boldsymbol{x}$ とすれば，この 2 クラス分類というのは，与えられた $\boldsymbol{x}$ が C_0 と C_1 のどちらに属するかを決定する作業であると言える．さらに数値的な変数 $y = 0, 1$ を導入すると，$\boldsymbol{x}$ をクラス C_y へ分類するということは，$\boldsymbol{x}$ の所属クラスを表す離散ラベル $y(\boldsymbol{x})$ の値を決めることであると言い換えられる．

$$\boldsymbol{x} \longrightarrow y(\boldsymbol{x}) \in \{0, 1\} \tag{3.2}$$

分類先のクラスが多数にわたる場合は多クラス分類と呼ばれる．例えば分類先を「仕事」「家族」「友人」... と複数のファルダに分ける場合，分類先が $C_1, C_2, C_3, \ldots, C_K$ となり，ラベル $y(\boldsymbol{x})$ も

1 から K の整数値をとる．

$$\boldsymbol{x} \longrightarrow y(\boldsymbol{x}) \in \{0, 1, \ldots, K\} \tag{3.3}$$

回帰

回帰 (regression) とは，データから，それに対応する実数値 (を並べたベクトル)$\boldsymbol{y}$ を予測作業のことである．例えば，過去数日の気象データ $\boldsymbol{x}$ から明日の気温を予測したとき，y は温度の数値に相当し，連続的な実数値をもつ変数になる．つまり，回帰とは，与えられた $\boldsymbol{x}$ を，対応する $\boldsymbol{y}$ に変換するために関数 $\boldsymbol{y}(\boldsymbol{x})$ を決定する作業である．

$$\boldsymbol{x} \longrightarrow \boldsymbol{y}(\boldsymbol{x}) \in \mathbb{R} \tag{3.4}$$

3.1.2 データセットの例

機械学習で学習を行う際，どのようなデータセットが用いられるのか．ここでは有名なデータセットである MNIST，CIFAR-10，ImageNet について紹介する．

MNIST

0 から 9 までの整数を機械に分類させるというタスクは機械学習の実装のチュートリアルとしてよく用いられる．このときに使われるデータベースが MNIST である．MNIST データベース (Mixed National Institude of Standards and Technology database) はアメリカの国立標準技術研究所 (NIST)，昔の国立標準局が提供した手書き数字のデータベースをシャッフルして作られたデータ集合である (図 3.2)．国勢調査局職員と高校生から集められた手書き数字の画像データが訓練用に 6 万枚，テスト用に 1 万枚用意されている．それぞれのサンプルはグレースケールの 28×28 ピクセル画像に整えられている．

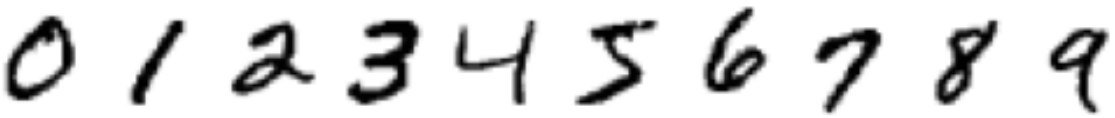


図 3.2: MNIST データセットの一部

CIFAR-10

自然画像の分類を機械学習で実装する際，チュートリアルとしてよく用いられるのが CIFAR-10 である．CIFAR-10 は 10 のカテゴリに分けられた 32×32 ピクセルの自然画像のデータセットである (図 3.3)．訓練用画像 6 万枚とテスト用画像 1 万枚からなる．

図 3.3: CIFAT-10 データセットの一部

ImageNet

ImageNet は，約 1400 万枚の自然画像からなる巨大なデータベースである．各画像に対して，写っている物体 1 つのカテゴリの正解ラベルがつけられており，クラス数は 2 万にも及ぶ．このデータベースは写真に写っている特定の物体を分類させる物体カテゴリ認識や，一般の画像に写っている物体を検出させて，さらにそれらを分類させる物体検出などのタスクを行う機械学習モデルを開発するために用意されたものである．

3.2 統計入門

機械学習とは，データ (経験) をもとにしてプログラムがいろいろなタスクをこなせるようにさせることであった．ではどのようにしたらプログラムはデータからタスクをこなすための知識を学びとれるか．それを理解するにはデータを科学的に分析する数理的手法である統計学の知識が必要である．ここでは，統計の基礎を確認し，どのようにして学習アルゴリズムや評価尺度を設計できるかについて説明する．

3.2.1 標本と推定

まず，データ (集合) やサンプル，標本という用語についてきちんと定めておく．これらはデータ点の集まりからなる．画像データの例でいえば，データというのは統計分析に用いるために用意した画像の集合のことで，1 枚 1 枚の画像のことをデータ点と呼ぶ．ただし，実際はデータ点も略してデータと呼んだり，さらにはサンプルをサンプルの要素であるデータ点の意味でも用いられたりする．つまり，これらは用語の乱用でされている．そのため，本論文でも文脈に応じてこの使い方に準じることにする．

データ点 (サンプル) の集まりからなるデータを分析するのが統計である．ここでは「推定」について考える．例として，日本人の平均身長の値を知りたい場合，これを正確に測るにはすべての日本人 (母集団) の身長を調べて，それの平均を求めればよい．しかし，それは現実的に不可能である．そのため，母集団の中から十分な数だけランダムに選び取った (抽出した) 日本人の身長データを集めて，その平均値をすべての日本人の平均身長であると推定するのである．つまり，「推定」というのは母集団からランダムに抽出[*1]されたデータのみを用いて分析し，母集団についての知識を獲得することを指すのである．

統計学では母集団の性質はデータ生成分布 $P_{\mathrm{data}}(x)$ により特徴づけられているものと仮定する．つまり，不確実性を伴う現象を確率的にモデル化するのである．これは我々の手にするデータは，自然界におけるさまざまな物理的過程の結果，この宇宙に存在することになったわけだが，我々がその過程すべてやそこに寄与する因子のすべてを知ることはできないため，データを何らかの確率論的な過程に従ってランダムに生じているとみなしているということである．例えば「サイコロを振ってどの目が出るか」という試行でも，もしサイコロやその周囲のすべての物理的情報を把握できるのであれば，原理的には出る目は力学で計算できるはずである．しかし人間には有限の認識・計算能力しかないため，どの目も 1/6 の確率で出るようにしか見えない．そのため，私たちはサイコロの目が出る確率は 1/6 と確率現象とみなしているのである．このように確率的にモデル化することでさまざまな現象を確率論的に予測することができる．

ここで，$\boldsymbol{x}$ という具体的なサンプルはデータ生成分布から抽出されたものであると仮定したとき，ある確率変数 x の実現値 x がデータ生成分布 $P(\mathrm{x})$ から抽出されたことを

$$\boldsymbol{x} \sim P_{\mathrm{data}}(\mathbf{x}) \tag{3.5}$$

と表す[*2]．

したがって母集団について知識を得るということは，データ生成分布を知ることを意味する．データを特徴づける十分な統計量をパラメータと呼ぶ．実際にはデータの生成の過程は極めて複雑なため，本当の $P(\mathrm{x})$ は無数のパラメータをもっており，分布を完全に知ることはほぼ不可能である．そこで通常は $P(\mathrm{x})$ をよく近似できると期待できるモデル分布 $P(\mathrm{x};\boldsymbol{\theta})$ を仮定し，そのモデルのパラメータ $\boldsymbol{\theta}$ の最適値 $\boldsymbol{\theta}^*$ をデータから推定する．これはパラメトリックはアプローチと呼ばれる．パラメトリックなモデルを仮定したことで，分布を特徴付ける少数のパラメータの値を推定すればよいことになる．このようにデータ生成の過程について推論できれば，その結果を使うことで新規のデータに関してもいろいろと予測することができる．

[*1] データの要素を 1 つ取り出すことは，正確には抜き取りという．

[*2] 抽出について 1 つ注意が必要である．それは，抽出されたデータは基本的に無作為抽出によって得られたもの．つまり，すべてのサンプルは同一分布から独立に抽出されたものとする点である．なぜなら，我々は母集団について知りたいため，サンプルの抽出方法によって，結果に偏りが生じてしまっては困るからである．

3.2.2 点推定

点推定とは，手持ちの有限個要素のデータ集合 $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ からパラメータの尤もらしい値を推定する手法である．点推定では，データを決める確率変数 $\{\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N\}$ の関数である推定量

$$\hat{\boldsymbol{\theta}}(\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N) \tag{3.6}$$

を作る．推定量そのものも確率変数であるため，データが具体的に与えられて，はじめてパラメータの推定値

$$\hat{\boldsymbol{\theta}}^*(x_1, x_2, \ldots, x_N) = \hat{\boldsymbol{\theta}}(x_1, x_2, \ldots, x_N) \tag{3.7}$$

を与えることになる．この推定値が，いま考えているパラメータをよく近似するように推定量を作れれば．データの数値から，背後にある分布について知ることができる．

よい推定量の作り方にはいくつもの指標がある．そこで，ここでは推定量に推奨される性質をいくつか紹介する．

バイアスが小さい

推定量の期待値 $\mathrm{E}\left[\hat{\boldsymbol{\theta}}\right]$ と真の値 $\boldsymbol{\theta}^*$ との差

$$b(\hat{\boldsymbol{\theta}}) = \mathrm{E}\left[\hat{\boldsymbol{\theta}}\right] - \boldsymbol{\theta}^* \tag{3.8}$$

のことをバイアスという．ここでの期待値は，データ生成分布での期待値なので，たくさんのデータ集合を用意して計算した平均値である．バイアスが小さいとは，推定量の偏りが小さく真の値から不要にズレていないということである．特にバイアスがゼロのものを不偏推定量と呼び，典型的な望ましい推定量である．一方でバイアスがあるものの，データの数が増えるにつれゼロへ漸近する状況 $\lim_{N\to\infty} b(\hat{\boldsymbol{\theta}}) = 0$ を漸近不偏推定量と呼ぶ．

分散が小さい

分散は，真の値に対してどれだけ推定量がばらつくかを測っている．したがって，これが小さいことは望ましいのは明らかである．

$$\mathrm{Var}\left(\boldsymbol{\theta}\right) = \mathrm{E}\left[\left(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\right)^2\right] \tag{3.9}$$

一致性

一致性とは，データ点の数を増えるにつれて統計量が真のパラメータに近づいていくという性質である．つまり，$N \to \infty$ に従い

$$\hat{\boldsymbol{\theta}} \to \boldsymbol{\theta}^* \tag{3.10}$$

となるということである．この性質を満たす推定量を一致推定量と呼ぶ．

ガウス分布の場合

点推定の例としてガウス分布を考える．ガウス分布とは実数値をとる確率変数 x 上の次のような分布である．

$$P(\mathrm{x}) = \mathcal{N}(\mathrm{x}; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mathrm{x}-\mu)^2}{2\sigma^2}} \tag{3.11}$$

この場合，$\boldsymbol{\theta} = \{\mu, \sigma^2\}$ であり，μ は平均，σ^2 は分散を表している．したがってこの 2 つのパラメータが決ればこの分布が定まる．では，ガウス分布から無作為に取り出した N 個のデータからパラメータ $\{\mu, \sigma^2\}$ を推定するにはどうすればよいか？

まず，μ の推定量 $\hat{\mu}$ について考える．今，どのサンプルも無作為に抽出されているとしたため，任意のデータ点 x_n はガウス分布に従う独立な確率変数である．したがって，その期待値は

$$\begin{aligned} E_{\mathcal{N}}[\mathrm{x}_n] &= \int_{-\infty}^{\infty} x_n P(x_n) dx_n \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} x_n e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx_n \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} (x'_n + \mu) e^{-\frac{x'^2_n}{2\sigma^2}} dx'_n \\ &= \frac{\mu}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{x'^2_n}{2\sigma^2}} dx'_n \\ &= \mu \end{aligned} \tag{3.12}$$

となり，分布のパラメータ μ に一致していることがわかる．そこで，μ の推定量 $\hat{\mu}$ を，与えられたデータ $\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ に関する平均値

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mathrm{x}_n \tag{3.13}$$

とする．そうすると，この推定量の期待値は

$$E_{\mathcal{N}}[\hat{\mu}] = \frac{1}{N} \sum_{i=1}^{N} \mathrm{E}[\mathrm{x}_n] = \mu \tag{3.14}$$

と見事に μ と一致し，不変推定量になっていることがわかる．

次に σ^2 の推定量はどう決めればよいか？ σ^2 の期待値を計算してみると

$$\begin{aligned} \mathrm{E}_{\mathcal{N}}[(\mathrm{x}_n - \mu)^2] &= \int_{-\infty}^{\infty} (x_n - \mu)^2 P(x_n) dx_n \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} x'^2_n e^{-\frac{x'^2_n}{2\sigma^2}} dx'_n \\ &= \sigma^2 \end{aligned} \tag{3.15}$$

となることから，安直には μ のときと同様に，サンプル平均

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathrm{x}_n - \hat{\mu})^2 \tag{3.16}$$

を用いればよいと思う．しかし，これの期待値を計算すると

$$\begin{aligned}
\mathrm{E}_{\mathcal{N}}\left[\hat{\sigma}^2\right] &= \mathrm{E}_{\mathcal{N}}\left[\frac{1}{N}\sum_{n=1}^{N}\left(\mathrm{x}_n - \hat{\mu}\right)^2\right] \\
&= \frac{1}{N}\mathrm{E}_{\mathcal{N}}\left[\sum_{n=1}^{N}\left\{\left(\mathrm{x}_n - \mu\right)^2 - 2\left(\mathrm{x}_n - \mu\right)\left(\hat{\mu} - \mu\right) + \left(\hat{\mu} - \mu\right)^2\right\}\right] \\
&= \frac{1}{N}\left\{\sum_{n=1}^{N}\mathrm{E}_{\mathcal{N}}\left[\left(\mathrm{x}_n - \mu\right)^2\right] - 2\sum_{n=1}^{N}\mathrm{E}_N\left[\left(\mathrm{x}_n - \mu\right)\left(\hat{\mu} - \mu\right)\right] + N\mathrm{E}_N\left[\left(\hat{\mu} - \mu\right)^2\right]\right\} \\
&= \sigma^2 - \frac{2}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathrm{E}_{\mathcal{N}}\left[\left(\mathrm{x}_n - \mu\right)\left(\mathrm{x}_m - \mu\right)\right] + \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathrm{E}_{\mathcal{N}}\left[\left(\mathrm{x}_n - \mu\right)\left(\mathrm{x}_m - \mu\right)\right] \\
&= \sigma^2 - \frac{1}{N^2}\sum_{n=1}^{N}\mathrm{E}_{\mathcal{N}}\left[\left(\mathrm{x}_n - \mu\right)^2\right] \\
&= \sigma^2 - \frac{1}{N}\sigma^2 \\
&= \left(1 - \frac{1}{N}\right)\sigma^2
\end{aligned}$$

となり，不偏推定量ではないという結果になる．ただし，$N \to \infty$ の極限でこの期待値は σ^2 に一致するため，漸近的不偏推定量になっている．余分な係数を除けば，バイアスがゼロになるため

$$\hat{\hat{\sigma}}^2 = \frac{1}{N-1}\hat{\sigma^2} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathrm{x}_n - \hat{\mu})^2 \tag{3.17}$$

とすれば不変推定量が得られる．

ベルヌーイ分布の場合

次にベルヌーイ分布の場合を考える．ベルヌーイ分布とはコイン投げの結果が表裏どちらかなど，2 つのランダムな値をとる現象を記述する分布である．確率変数を 0 か 1 の離散値をとるものとし，x が 1 をとる確率を p とすると，この分布は，

$$P(\mathrm{x}) = p^{\mathrm{x}}(1-p)^{1-\mathrm{x}} \tag{3.18}$$

と書ける．このときパラメータは p のみである．ベルヌーイ分布の期待値と分散は，

$$\mathrm{E}_P[\mathrm{x}] = \sum_{\mathrm{x}=0,1} xP(x) = 1 \cdot P(1) = p \tag{3.19}$$

$$\begin{aligned}\mathrm{E}_P[(\mathrm{x}-p)^2] &= \sum_{\mathrm{x}=0,1}(x-2px+p^2)P(x)\\ &= p^2\cdot P(0)+(1-2p+p^2)\cdot P(1)\\ &= p(1-p)\end{aligned} \tag{3.20}$$

であるため，パラメータの推定量は再びサンプル平均

$$\hat{p}=\frac{1}{N}\sum_{n=1}^{N}x_n \tag{3.21}$$

がよさそうである．実際,

$$\mathrm{E}_P[\hat{p}]=\frac{1}{N}\sum_{N=1}^{n}E_P[\mathrm{x}_n]=p$$

となり，不変推定量であることがわかる．

では，分散はどうなるか？期待値を計算してみると

$$\mathrm{E}\left[(\hat{p}-p)^2\right]=\frac{1}{N^2}\sum_{n=1}^{N}\mathrm{E}_P\left[(x_n-p)^2\right]=\frac{1}{N}p(1-p) \tag{3.22}$$

となり，データサイズが大きくなるほど推定値のばらつきがゼロに近づいていく，したがって大きなデータに対して分散は小さくなるような推定量である．

3.2.3 最尤推定

点推定では発見的な方法で推定量を探してきたが，これでは複雑な分布の場合に推定量を見つけることができない．ところがパラメトリックな場合には，実は広く使える強力な手法がある．それが最尤推定法である．

データ生成分布のパラメトリックモデル $P_{\mathrm{model}}(\mathrm{x};\boldsymbol{\theta})$ が与えられているとする．与えられたサンプル $\mathcal{D}=\{x_1,x_2,\ldots,x_N\}$ は，この分布から無作為に抽出されていると近似する．すると，これらは独立に同じモデルから生成された実現値であるため，このデータ集合が得られる同時確率は

$$P(x_1,x_2,\ldots,x_N;\boldsymbol{\theta})=\prod_{n=1}^{N}P_{\mathrm{model}}(x_n;\boldsymbol{\theta})$$

である．これを変数 $\boldsymbol{\theta}$ に対する量とみなして $L(\boldsymbol{\theta})=P(x_1,x_2,\ldots,x_N;\boldsymbol{\theta})$ と書き，尤度または尤度関数と呼ぶ．データ値が $\{x_1,x_2,\ldots,x_N\}$ という観測値を取ったのは，パラメータ $\boldsymbol{\theta}$ の値が確率 $P(x_1,x_2,\ldots,x_N;\boldsymbol{\theta})$ を大きくするようなものであったからであると考える．つまり，$L(\boldsymbol{\theta})$ を最大にするようなパラメータ値であるためにデータ $\{x_1,x_2,\ldots,x_N\}$ が実現されやすかったと解釈するのである．すると，与えられたデータに対して尤もらしいパラメータの値は尤度を最大化したものということになる．

$$\boldsymbol{\theta}_{ML}=\underset{\boldsymbol{\theta}}{\mathrm{argmax}}L(\boldsymbol{\theta}) \tag{3.23}$$

ただし，尤度は確率の積であるため，1 以下の数値を何回も掛け合わせた値になる．値が小さすぎると計算機上でアンダーフローを起こしうるため，実用上，対数尤度を最大化する．

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}(\ln L(\boldsymbol{\theta})) \tag{3.24}$$

尤度と対数尤度のどちらで最大化を行っても結果は変わらないことか明らかである．機械学習ではこのように何らかの目的関数を最大化，もしくは最小化することで推定に用いる最適なパラメータ値を決定するのがほとんどである．実際，多くの機械学習アルゴリズムは最尤法を基礎に構築される．ただし，実際の機械学習では，最大値よりも最小値をして問題を書く場合が多いため，負の対数尤度を目的関数とした最小化問題として表現する．

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(-\ln L(\boldsymbol{\theta})) \tag{3.25}$$

ガウス分布の場合

ガウス分布で最尤推定法を考える．N 個のデータに対する尤度は

$$L(\boldsymbol{\theta}) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_n-\mu)^2}{2\sigma^2}} \tag{3.26}$$

である．ただし，$\boldsymbol{\theta} = (\mu, \sigma^2)$ とした．対数尤度は

$$\ln L(\boldsymbol{\theta}) = -\frac{N}{2}\ln\sigma^2 - \sum_n (x_n - \mu)^2 + \text{const.} \tag{3.27}$$

となる．この関数の最大値を探すには，微分係数がゼロの場所を求めればよいので，

$$\left.\frac{\partial \ln L(\boldsymbol{\theta})}{\partial \mu}\right|_{\boldsymbol{\theta}_{ML}} = \frac{1}{\sigma^2_{ML}} \sum_{n=1}^{N} (x_n - \mu_{ML}) = 0 \tag{3.28}$$

$$\left.\frac{\partial \ln L(\boldsymbol{\theta})}{\partial \sigma^2}\right|_{\boldsymbol{\theta}_{ML}} = -\frac{N}{2\sigma^2_{ML}} + \frac{1}{2\left(\sigma^2_{ML}\right)^2} \sum_{n=1}^{N} (x_n - \mu_{ML})^2 = 0 \tag{3.29}$$

を解けばよい．これはすぐに解け，得られる推定量は

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{3.30}$$

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu_{ML})^2 \tag{3.31}$$

となる．これは推定量 $(\hat{\mu}, \hat{\sigma}^2)$ での推定値と等しくなっている．

ベルヌーイ分布の場合

ベルヌーイ分布に対しては，対数尤度は

$$L(p) = \prod_{n=1}^{N} p^{x_n}(1-p)^{1-x_n} \tag{3.32}$$

となる．よって対数尤度は

$$\ln L(p) = \sum_{n=1}^{N}(x_n \ln p + (1-x_n)\ln(1-p)) \tag{3.33}$$

となる．これの微分係数がゼロの場所は

$$\begin{aligned}\left.\frac{\partial \ln L(p)}{\partial p}\right|_{p_{ML}} &= \sum_{n=1}^{N}\left(\frac{x_n}{p} - \frac{1-x_n}{1-p}\right)\Bigg|_{p_{ML}} \\ &= \left(\frac{\sum_n x_n - N p_{ML}}{p_{ML}(1-p_{ML})}\right)\end{aligned} \tag{3.34}$$

を解けばよい．したがって

$$p_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{3.35}$$

となり，これもまた先ほど求めた推定量 $\hat{p}$ の推定値と一致している．このように最尤法は推定量を求めるための汎用性のある方法である．

3.3 機械学習の基礎

この章の 1 節で機械学習とは何かについて定義し，大まかな学習の流れを説明した．ここでは機械学習についてもう少し踏み込んだ内容について詳しく説明していく．

3.3.1 教師あり学習

機械学習にはいくつもの枠組が存在しているが，大きく「教師あり学習」,「教師なし学習」,「強化学習」の 3 つに分類される．ここでは教師あり学習について説明する．

まず,「教師あり」と「教師なし」の違いは何かというと，学習に用いる訓練データが前者は入力値と正解値のペアになっており，後者は入力値のみになっているという点である．したがって，教師あり学習では用意すべき訓練データが必ず入力 $\boldsymbol{x}$ と出力 $\boldsymbol{y}$ のペアの形をとっている．

$$\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} \tag{3.36}$$

この入力と出力の間の関係を推定することで，新しい未知の $\boldsymbol{x}$ が与えられたときに対応する出力 $\boldsymbol{y}$ を適切に予測できるようになることが教師あり学習での目的である．

$$\boldsymbol{x} \longrightarrow \boldsymbol{y} \tag{3.37}$$

統計学の用語を用いていえば，説明変数 $\mathbf{x}$ と目標変数 $\mathbf{y}$ の間の関係式を知り，常に目標変数を予測できるモデルを作ろうということである．そのために目標変数[*3]の適切な推定量 $\hat{y}$ を探すのが学習である．実際には推定量の関数モデルを仮定して，そのパラメータを訓練データから最適化する学習アルゴリズムを実装する．

3.3.2 最小二乗法による線形回帰

教師あり学習における回帰問題の手法として最小二乗法を用いた線形回帰について説明する．ここでは出力がベクトルではなくスカラーの場合を考える．つまり，入力 $\boldsymbol{x}$ から出力値 y を予想するという回帰問題を扱うということである．

まず，パラメトリックなモデルを考える．つまり，すべてのパラメータをまとめて $\boldsymbol{w}$ としたとき，y は常にある関数 $\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$ で表される規則で $\boldsymbol{x}$ が決まっているとする．このとき，パラメータ $\boldsymbol{w}$ の値はまだ未知であり，与えられたデータから最適なパラメータ値を決める作業を行う．これが学習に相当する．また，データの観測には必ずノイズが生じるため，それを確率変数 ϵ として，

$$y = f(\boldsymbol{x}; \boldsymbol{w}) + \epsilon \tag{3.38}$$

のようにモデル化する．その一方で我々が知りたいのは $\boldsymbol{x}$ と y の間の対応規則を捉える項 $f(\boldsymbol{x}; \boldsymbol{w})$ である．

ここで考えるのは，特に規則性の部分がパラメータの 1 次関数であることを仮定した回帰モデルである．

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{h}(x) = \sum_{j=0} w_j h_j(\boldsymbol{x}) \tag{3.39}$$

これはパラメータに対して線形であるため線形回帰と呼ばれる．一方で $h_j(\boldsymbol{x})$ は必ずしも線形関数である必要はない．例えば，入力 x がベクトルではなくスカラーの場合，$h_j(x) = x^j$ という単項式を選んでもよい．

$$f(x; \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{h}(x) = \sum_{j=0}^{M} w_j x^j \tag{3.40}$$

これは多項式回帰とよばれ，多項式回帰も線形回帰の一種である．

回帰分析ではモデル $f(\boldsymbol{x}; \boldsymbol{w})$ が与えられたデータ $\mathcal{D}$ の入出力が満たす対応関係によく当てはまるように，モデルのパラメータ $\boldsymbol{w}$ を調整する．そのためにはモデルがどの程度データに当てはまっているかを測る尺度が必要である．機械学習では特に，誤差関数または損失関数と呼ばれるモデル $f(\boldsymbol{x}; \boldsymbol{w})$ をパフォーマンスの悪さを測る尺度を最小化することにより最適パラメータ $\boldsymbol{w}^*$ を決

[*3] 目標変数には，離散値ととる確率変数である質的変数と，連続値をとる確率変数である量的変数の 2 種類がある

定する．誤差関数にはさまざまな種類があるが，その中で最も有名で理解しやすいのが，次のような平均二乗誤差である．

$$E_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}(\boldsymbol{x}_n;\boldsymbol{w}) - y_n)^2 \tag{3.41}$$

$\hat{y}(\boldsymbol{x}_n;\boldsymbol{w})$ はモデルに n 番目の入力データを入れて得られるモデル予測であり，y_n が実際にデータに対応した出力値である．したがって，$(\hat{y}(\boldsymbol{x}_n;\boldsymbol{w}) - y_n)^2$ は予測と正解の差を 2 乗したものである．これはまさに予測と正解のズレを測っており，それを全サンプルについて平均したものがこの誤差関数である．そして，平均二乗誤差を最小化してパラメータ最適化を行う手法が平均二乗法である．

平均二乗法とは平均二乗誤差を最小化するパラメータの探索であるため，次のような最小値問題を解くことに対応する．

$$\boldsymbol{w}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(E_{\mathcal{D}}(\boldsymbol{w})) \tag{3.42}$$

これを解くことで，関数 $\hat{y}$ から正しい出力値を得ることができるようになる．

ここで誤差関数の最小化によるパラメータ最適化を行う上で知っておきたい重要な概念を述べておく．機械学習の本来の目的は，手持ちの訓練データだけでなく，母集団すべてデータに対してよい良い出力をするモデルを作ることである．訓練データをもとにしてアーキテクチャを訓練し，訓練データにない未知のデータに対してまでよく働く状況が実現されることを汎化という．汎化を達成するために本来最小化すべき誤差関数は本来は訓練誤差ではなく，次のような，モデルの出力と実際の正解値との差の 2 乗をデータ生成分布による期待値で定義した汎化誤差なのである．

$$E_{\mathrm{gen}}(\boldsymbol{w}) = \mathrm{E}_{(\mathrm{x},\mathrm{y})\sim P_{\mathrm{data}}}\left[(\hat{\mathrm{y}}(\mathbf{x};\boldsymbol{w}) - y)^2\right] \tag{3.43}$$

しかしながら，有限な能力しかない我々には，可能なすべてのデータを集めてくることはできない．したがって，汎化誤差を最小化を行うことは不可能なのである．そこで機械学習では手持ちのデータのサンプル平均で汎化誤差を近似的に見積った，訓練誤差を最小化しているのである．訓練誤差を小さくしただけでは，汎化がそう易々と実現するとはなかなか言えない．しかし驚くべきことに，訓練誤差の最小化でも，汎化を実現することは可能なのである．

3.3.3 学習不足と過学習

訓練誤差の最小化によって汎化を実現することは可能であることを述べたが，汎化の実現がそう簡単ではない．モデルのパラメータ数の設定値によって，学習不足や過学習といった汎化を妨げる現象が起こる．ここでは，学習不足と過学習について説明する．

簡単のために 1 次元データの多項式回帰を考える．つまり，与えられた入力 x から出力 y を予測する規則性を推定するために，多項式モデル

$$\hat{y} = \sum_{j=0}^{M} w_j x^j \tag{3.44}$$

をデータにフィッティングする．この際，多項式の次数 M の値は我々が選ばなくてはならない．このようにモデルの選択時に決めなければならないパラメータをハイパーパラメータと呼び，重みやバイアスといった学習されるパラメータと区別する．例として，$M = 2$ で設定した場合，

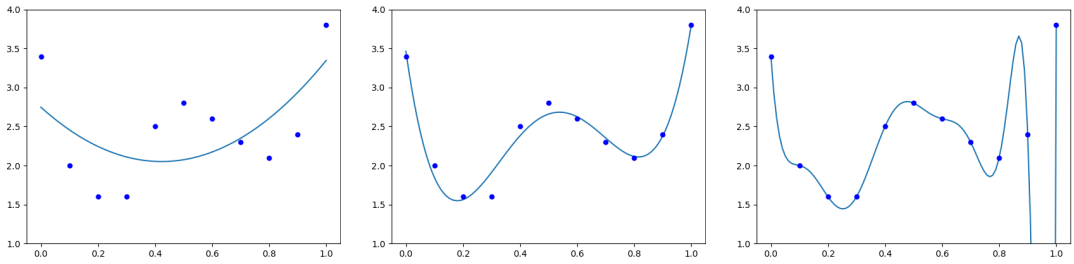
$$\hat{y} = w_2 x^2 + w_1 x + w_0 \tag{3.45}$$

となり，これは 2 次関数でフィッティングするということである．このような多項式モデルの場合，M を設定することで自動的に重みパラメータの個数も決定しているため，まさに M がモデルの自由度を与えていることが分かる．図 3.4 は 1 次元データに対して $M = 2, 4, 20$ でフィッティングした結果である．$M = 2$ に対応する図 3.4(a) のように，データに対してモデルの自由度が小さすぎると，データの構造を捉えることはまったくできない．つまり，訓練誤差の値が大きすぎて，これでは何の予測能力の得られないだろう．このような状況は学習不足またはアンダーフィッティングと呼ばれる．

M を大きくしていけば，データの複雑な構造をよりよく捉えられるようになる．しかし，不必要に大きな自由度をもったモデルは，異なる問題を引き起こす．というのも，モデルの自由度が大きすぎると，学習データのもつノイズ (統計的なゆらぎ) までも多項式モデルが正確にフィッティングしてしまうのである．図 3.4(c) は $M = 20$ でフィッティングを行った結果であり，ちょうどそのような状況になっている．与えられた訓練データに関する誤差関数に値は確かにどんどん小さくできるが，未知のデータに対してはどんどん予測能力を失っていく．図の右側はそれが顕著に表れていることがわかる．この状況は過学習またはオーバーフィッティングと呼ばれる．

このようにモデルの自由度が小さすぎると学習不足になり，逆に大きすぎると過学習になってしまうため，我々はこの間にあるちょうどよい自由度のモデルを設定する必要場ある．図 (3.4)(b) は $M = 4$ でフィッティングした結果であるが．これはモデルの自由度と学習すべきデータを含む情報の豊かさがつり合っており汎化が実現していそうな良い状況であることが見てわかる．

このように機械学習をうまくいかせるためには，ちょうどよいモデルパラメータの数，つまりモデルの自由度を見積もらなくてはならない．自由度の見積もりにはいろいろな基準があるが，残念なことに深層学習のような複雑なモデルではどれもあまり役立たないのが現状である．そこで実務の場面では，学習に用いるデータとは別に検証用のデータを用意して検証用データに関する誤差を目安に，仮定したモデルがよいものかどうかを見極めることになる．



(a) 学習不足 **(b)** 適切なパラメータ数 **(c)** 過学習

図 3.4: 1 次元データの線形回帰の例

3.3.4 正則化

過学習を避けるための一番わかりやすい方法は，自由度の数が大きすぎないモデルをはじめから選んでおくことである．自由度を実質的に減らしてしまうさまざまな手法は正則化と呼ばれる．実はモデル自体を修正することなく，学習アルゴリズムを少し変更するだけで実質的な自由度を減らすことができる．一般的にこのような正則化は，最適化すべき誤差関数の変更として表現することができる．

$$E_{\mathrm{new}}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda R(\boldsymbol{w}) \tag{3.46}$$

λ は正則化の効果の大きさを調整するための正則化パラメータである．

重み減衰 (weight decay) はこのような正則化の代表例である．多項式回帰の例からわかるように，もし強制的に多くのパラメータの値を 0 に制限することができれば，はじめから調整できるパラメータが少ないため自由度を減らしたことと同じことになる．そこで最小化する誤差関数に，次のような項を加えてみる．

$$E_{\mathrm{wd}}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda \boldsymbol{w}^{\top}\boldsymbol{w} \tag{3.47}$$

新たに項に加わった右辺全体を最小化するということは，重みベクトルのノルム $\boldsymbol{w}^{\top}\boldsymbol{w}$ をできる限り小さくする解 $\boldsymbol{w}^{*}$ がより好まれるようになるということである．そのため，このように修正された後の最適解では，可能な限りの $\boldsymbol{w}$ の成分がほぼゼロになっている．したがって重み減衰はモデル自体を変えずに，最小化する誤差関数へのパラメータ数を減らすペナルティ項を加える正則化である．回帰に重み減衰を適用したものは Ridge 回帰と呼ばれる．

Ridge 回帰以外にも，さまざまな回帰の正則化が存在する．その一例は

$$E_{\mathrm{wd}}(\boldsymbol{w}) = E(\boldsymbol{w}) + \lambda \sum_{i} |w_i| \tag{3.48}$$

という誤差関数と用いる LASSO 回帰である．この手法もまた，不要な重みをできるだけゼロへ近づける効果がある．

与えられたタスクのパフォーマンス向上という目標は維持しつつも，モデルの自由度を減らして過学習を避ける正則化の手法は，汎化性能の実現を目指す機械学習では極めて重要な位置を占める．

3.3.5 クラス分類

これまで教師あり学習の回帰問題を扱ってきた．ここでは，分類問題について考える，クラス分類とは与えられた入力 $\boldsymbol{x}$ を K 個のクラス $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ へ分類する作業である．これらのクラスは互いに排他的であり，1 つの入力に対して必ず 1 つの所属先があるものとする．回帰のように数値的に扱うために，離散的な目標変数を導入する．

2 値分類の場合

分類先が 2 つしかない状況では，1 つ目のクラスに属していることを $y = 1$，2 つ目のクラスに属していることを $y = 0$ として表現する 2 値変数 y を用いればよい．

多クラス分類の場合

手書き数字 $0 \sim 9$ のどれかを判定する場合など，分類先クラスが複数ある場合はいろいろなアプローチがある．最もシンプルなものは，K 個のクラスそれぞれに対応して $y = 1, 2, \ldots, K$ という K 個の値をとる離散変数を用いる方法である．手書き数字の例では $K = 10$ である．この変数を，ベクトルを用いた別の表現である，1-of-hot 符号化に写像できる．y によって決まる K 成分ベクトル

$$\boldsymbol{t}(y) = \begin{pmatrix} t(y)_1 & t(y)_2 & \cdots & t(y)_K \end{pmatrix}^\top \tag{3.49}$$

が 1-of-hot 符号化である．この手法では入力が k 番目のクラスに属している．つまり，$y = k$ であることを，第 k 成分が $t(y = k)_k = 1$ でそれ以外の成分 $t(y = k)_{l \neq k}$ はすべて 0 であることで表現している．このような表現法を one-hot 表現と呼ぶ．例えば一番のクラスに属しているのであれば，

$$\boldsymbol{t}(y = 1) = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}^\top \tag{3.50}$$

ということである．ここで，クロネッカーのデルタ記号

$$\delta_{i,j} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \tag{3.51}$$

を使うと，one-hot 表現ベクトルの成分は

$$t(y)_k = \delta_{y,k} \tag{3.52}$$

と書くことができる．

3.3.6 クラス分類へのアプローチ

クラス分類では与えられた入力に対し，離散的な目標変数を予測したいわけだが，そのためには複数のアプローチが考えられる．

関数モデル

関数モデルは，入力と出力の間の関係を関数としてモデル化する方法である．

$$\hat{y} = y(\boldsymbol{x}; \boldsymbol{w}) \tag{3.53}$$

例えば 0,1 の 2 値をとる目標変数に対し，

$$y(\boldsymbol{x}; \boldsymbol{w}) = f(\boldsymbol{w}^\top \boldsymbol{x} + b) \tag{3.54}$$

というモデルを仮定すると，これは 1 層ニューラルネットワークになっている．

生成モデル

関数モデルによるアプローチ以外の手法では，データを確率的に取り扱う．生成モデルではデータに潜むランダム性を同時分布 $P(\boldsymbol{x}, \boldsymbol{y})$ のモデル化により表現し，このモデルをデータにフィットさせる．k 成分だけが 1 の $\boldsymbol{t}(\boldsymbol{y})^\top = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ に対する $P(\boldsymbol{x}, \boldsymbol{y})$ は，入力が $\boldsymbol{x}$ で所属クラスが $\mathcal{C}_k$ である確率を意味する．この代わりに $P(\boldsymbol{x}, \mathcal{C}_k)$ をモデル化しても同じことである．

この確率をすべてのクラスについて求めたあと，$\boldsymbol{x}$ について周辺化して $P(\boldsymbol{x})$ を求めてベイズの公式を使うことで条件付き確率 $P(\mathcal{C}_k|\boldsymbol{x})$ を求める．$P(\mathcal{C}_k|\boldsymbol{x})$ がわかれば任意のデータ $\boldsymbol{x}$ がわかれば任意のデータ $\boldsymbol{x}$ が各クラスに属している確率が評価でき，得られた確率の値がもっとも大きくなるクラス $\mathcal{C}_k$ を $\boldsymbol{x}$ の所属先と予測することになる．

識別モデル

生成モデルではまず同時分布を得て，そこから条件付き確率分布 $P(\mathcal{C}_k|\boldsymbol{x})$ を計算していた．このような回りくどいことをせずに $P(\mathcal{C}_k|\boldsymbol{x})$ おモデル化してデータに学習させるのが認識モデルである．予測する方法は生成モデルと同様である．

3.3.7 ロジスティック回帰

クラス分類を例にとって，識別モデルをもう少し詳しく見ていく．まずは 2 クラス分類から考える．この場合はクラスが 2 つしかないため，確率分布は $P(\mathcal{C}|\boldsymbol{x}) = P(\boldsymbol{x}|\mathcal{C}) = 1$ を満たす．ここでシグモイド関数

$$\sigma(u) = \frac{1}{1 + e^{-u}} = \frac{e^u}{1 + e^u} \tag{3.55}$$

を導入する．今考えていた，条件付き確率は

$$P(\mathcal{C}_1|\boldsymbol{x}) \tag{3.56}$$

とシグモイド関数で書くことができる．ここで，u は対数オッズと呼ばれる因子で

$$u = \ln \frac{P(\mathcal{C}_1|\boldsymbol{x})}{1 - P(\mathcal{C}_1|\boldsymbol{x})} = \ln \frac{P(\mathcal{C}_1|\boldsymbol{x})}{P(\mathcal{C}_2|\boldsymbol{x})} \tag{3.57}$$

である．

$$e^u = \frac{P(\mathcal{C}_1|\boldsymbol{x})}{P(\mathcal{C}_2|\boldsymbol{x})} \tag{3.58}$$

がオッズ比と呼ばれ，$\mathcal{C}_1$ である確率とそうでない確率の比率を表す．対数オッズが 0 を超えると $\mathcal{C}_1$ である確率が $\mathcal{C}_2$ である確率を上回ることになる．したがって対数オッズも明快な意味をもつため，確率モデルを設計する際に基本的な道具となる．

実際多くのクラス分類では，対数オッズ u が入力に関する線形関数であることを仮定した単純な確率モデルが用いられる．

$$u = \boldsymbol{w}^\top \boldsymbol{x} + b \tag{3.59}$$

$\boldsymbol{w}$ はモデルのパラメータをまとめて表したベクトルである．このモデルは統計分析においてロジスティック回帰と呼ばれ，ベルヌーイ分布の統計分析法として広く用いられている．また，ロジスティック回帰は，一般化線形モデルという重要な統計モデルの典型例である．

訓練データとのフィッティングには最尤法を用いる．これもまた，一般化線形モデル全般に通じる方法である．まず離散数値をとる目標変数 y は，$y = 1$ が 1 つ目のクラス $\mathcal{C}_1$ に入っている場合，$y = 0$ が $\mathcal{C}_2$ に入っている場合を表していた．つまり，

$$P(y = 1|\boldsymbol{x}) = P(\mathcal{C}_1|\boldsymbol{x}), \quad P(y = 0|\boldsymbol{x}) = 1 - P(\mathcal{C}_1|\boldsymbol{x}), \tag{3.60}$$

である．$P(y|\boldsymbol{x})$ はベルヌーイ分布なので，一般的に次のように書ける．

$$P(y|\boldsymbol{x}) = (P(\mathcal{C}_1)|\boldsymbol{x})^y \left(1 - P(\mathcal{C}_1)|\boldsymbol{x}\right)^{1-y} \tag{3.61}$$

これは右辺に $y = 1$ と $y = 0$ を実際に代入すればすぐに理解できる．したがって，これをデータ $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)\}$ によって学習させるには，最尤法に従って，次の尤度関数

$$L(\boldsymbol{w}) = \prod_{n=1}^{N} (P(\mathcal{C}_1)|\boldsymbol{x}_n)^{y_n} \left(1 - P(\mathcal{C}_1)|\boldsymbol{x}_n\right)^{1-y_n} \tag{3.62}$$

を最大化すればよかった．ここでロジスティック回帰のパラメータ $\boldsymbol{w}$ は，対数オッズの部分を線形関数でモデル化した際に式 (**??**) で挿入されたパラメータであったことを思い出しておこう．機械学習の実装で用いられるのは対数尤度であったので，負の対数尤度で誤差関数を定義する．

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N} \left(y_n \ln P(\mathcal{C}_1|\boldsymbol{x}_n) + (1 - y_n) \ln \left(1 - P(\mathcal{C}_1|\boldsymbol{x}_n)\right)\right) \tag{3.63}$$

これはデータの経験分布とモデル分布の間の交差エントロピーと呼ばれ，深層学習でもよく登場する重要な誤差関数の 1 つである．

3.3.8 ソフトマックス回帰

多クラス分類の識別モデルについて考える．この場合もロジスティック回帰の一般化で取り扱うことができる．次の式に注目する．

$$P(y|\boldsymbol{x}) = \prod_{k=1}^{K} (P(\mathcal{C}_k|\boldsymbol{x}))^{t(y)_k} \tag{3.64}$$

これが正しい式であることは，one-hot 表現の定義である式 (**??**) からわかる．例えば，これは $\boldsymbol{t}(y) = \begin{pmatrix} 1 & 0 & 0 & \cdots \end{pmatrix}^\top$ に対応した y を考えたならば，式 (**??**) の右辺は $P(\mathcal{C}|\boldsymbol{x})$ となるからである．この分布はベルヌーイ分布の多値変数への自然な拡張である，マルチヌーイ分布やカテゴリカル分布と呼ばれる．この分布を少し書き換えてみると，$\sum_{k=1}^{K} t(y)_k = 1$ が常に成り立っているため，

$$\begin{aligned} P(y|\boldsymbol{x}) &= \prod_{k=1}^{K-1} (P(\mathcal{C}_k|\boldsymbol{x}))^{t(y)_k} (P(\mathcal{C}_K|\boldsymbol{x}))^{1-\sum_{k=1}^{K-1} t(y)_k} \\ &= (P(\mathcal{C}_K|\boldsymbol{x})) \prod_{k=1}^{K-1} \left(\frac{P(\mathcal{C}_k|\boldsymbol{x})}{P(\mathcal{C}_k|\boldsymbol{x})} \right)^{t(y)_k} \\ &= (P(\mathcal{C}_K|\boldsymbol{x})) \prod_{k=1}^{K-1} e^{t(y)_k u_k} \\ &= (P(\mathcal{C}_K|\boldsymbol{x})) e^{\sum_{k=1}^{K} t(y)_k u_k} \end{aligned} \tag{3.65}$$

が得られる．u_k は対数オッズを多クラス問題に一般化した

$$u_k = \ln \frac{P(\mathcal{C}_k|\boldsymbol{x})}{P(\mathcal{C}_k|\boldsymbol{x})} \tag{3.66}$$

である．

このように多クラス分類でも，対数オッズは分布を記述するためのよいパラメータである．対数オッズの定義からただちに $P(\mathcal{C}_K|\boldsymbol{x})e^{u_k} = P(\mathcal{C}_k|\boldsymbol{x})$ が得られるが，右辺の和を取ると全確率の法則から $\sum_k P(|||\boldsymbol{x}) = 1$ なので，

$$P(\mathcal{C}_K|\boldsymbol{x}) = \frac{1}{\sum_{k=1}^{K} e^{u_k}} \tag{3.67}$$

という関係式が得られる．これを元の式 $P(\mathcal{C}_K|\boldsymbol{x})e^{u_k} = P(\mathcal{C}_k|\boldsymbol{x})$ に入れ直すことで，各クラスの分布をソフトマックス関数で書き表すことがわかった．

$$P(\mathcal{C}_k|\boldsymbol{x}) = \mathrm{softmax}_k(u_1, u_2, \ldots, u_K) = \frac{e^{u_k}}{\sum_{k'=1}^{K} e^{u_{k'}}} \tag{3.68}$$

このように，カテゴリカル分布は対数オッズを引数とするソフトマックス関数で書き表すことができた．そこで，ロジスティック回帰に倣い，その対数オッズを線形関数でモデル化する．

$$u_k = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k, \quad (k = 1, 2, \ldots, K-1) \tag{3.69}$$

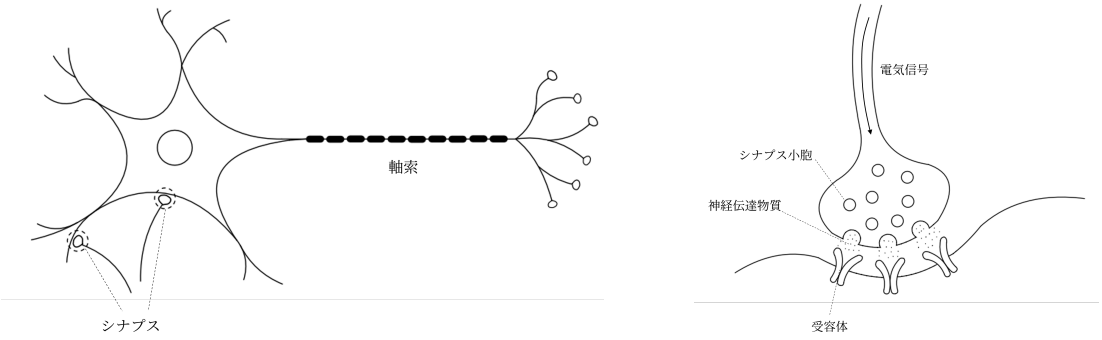
この線形の対数オッズを用いた識別モデルをソフトマックス回帰と呼ぶ．ソフトマックス回帰を最尤法で取り扱う方法も，ロジスティック回帰の場合とまったく同じである．つまり，負の対数尤度関数

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{K-1}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t(y_n) \ln P(\mathcal{C}_k|\boldsymbol{x}_n) \tag{3.70}$$

を最小化することでパラメータを最適化する．ただし，$\sum_k P(\mathcal{C}_k|\boldsymbol{x}_n) = 1$ と $\sum_k t(y_n)_k = 1$ という条件が付いていることに注意である．そのため余分な 1 自由度が消去され，式 (**??**) により，$u_K = 0$ となっている．したがって，K 番目のクラスにはパラメータ $\boldsymbol{w}_K$ が付与されていない．

3.4 ニューラルネットワーク

3.4.1 神経細胞のネットワーク



(a) ニューロンの概要図　　**(b)** シナプスの概要図

図 3.5: ニューロンとシナプス

ニューラルネットワークは人間の脳の神経回路を模した数理モデルである．ニューラルネットワークの構造や内部の仕組みについて理解するために，まずは人間の脳内での情報伝達の仕組みについて概要を説明する．

人間の脳は，1000 億個以上ものニューロンと呼ばれる神経細胞が寄り集まって構成されている．ニューロンの形状は 1873 年に C. ゴルジが銀とクロムを用いた細胞の染色法によって観察に成功したことで明らかになった．図 3.5(a) の形状を簡単に表した図である．ニューロンは通常の細胞とは大きく異なり，約 10μm ほどの大きさの細胞体と呼ばれる部分から何本もの突起が伸びた構造をしている．このうち 1 本ある細長い突起は軸索と呼ばれ，その長さはヒトの場合，1mm から 1m である．軸索の先端付近ではいくつもの分岐しており，これらを軸索側枝と呼ぶ．側枝はせいぜい数十 μm ほどの長さしかない．そして側枝の終端を軸索終末と呼ぶ．軸索以外の突起は樹状突起と呼ばれる．樹状突起は植物の根のような途中でいくつも枝分かれした形状をしており，樹状突起の全長はせいぜい数 mm と，軸索と比べるとはるかに短い．

このように突起がたくさん飛び出したニューロンがたくさん集まり，それらが規則的に接合を繰

り返して脳はできている．図 3.5(a) の丸で囲まれた部分がニューロン同士が接合する部分でこれをシナプスと呼ぶ．シナプスは軸索終末と樹状突起，または細胞体と接合して形成される．シナプスを拡大したものが図 3.5(b) である．

では，軸索，樹状突起，そしてシナプスはどのような働きをしているかといいうと，まず軸索も樹状突起も，電気信号 (パルス) を伝える電線のような役割を果たしている．神経回路上の電気信号は，図**??**のように極めて短い時間だけ立ち上がるパルスとして伝播する．このパルスの振幅は決まっているため，パルスの波の高さが信号の大きさを決めるわけではない．パルスは短い時間に細かく密集して伝わるが，信号強度に対応するのはそのパルスの密度である．

シナプス部ではまず，軸索に伝わってきた電気信号を合図に，軸索終末がシナプス小胞というカプセルに詰め込まれた化学物質を外にばらまく．その化学物質は神経伝達物質とよばれ，放出後はシナプスの樹状突起側で受け止められる．この場所には多数のレセプターがあり，そこに神経伝達物質が結合すると，その刺激から新たな電気信号が生み出される仕組みになっている．シナプスは樹状突起上にたくさんあり，各シナプスで樹状突起は他のさまざまなニューロンから電気信号の出力を受け入れる．この電気信号は細胞体へと向かって伝播する．そして数多くの樹状突起から伝わってきた電気信号は，細胞体に到達しすべて合算される．

細胞体は，ある一定以上の大きさ，つまり閾値を越える電気信号を受けると，軸索に向かって電気信号を出力する．電気信号は各軸索終末にあるシナプスにおいて他のニューロンの樹状突起に入力し，同じ伝播のパターンを繰り返していく．このように相互作用を複数に繰り返すことで，神経細胞のネットワークはできている．

3.4.2 形式ニューロン

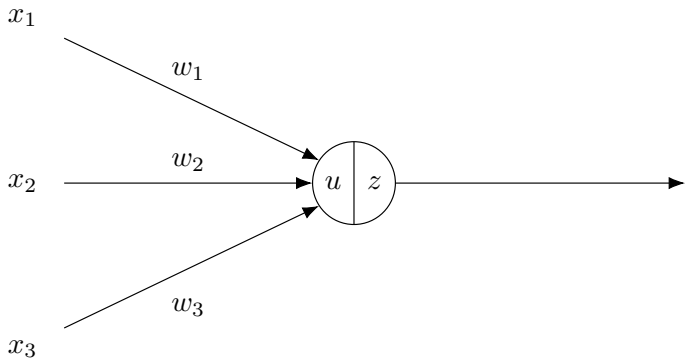


図 3.6: 形式ニューロンの例

1943 年に W. マカロックと W. ピッツによってニューロンの活動を数理論理学的な手法でモデル化できることを発見し，神経活動の数理モデルと論理回路との対応を明らかにした．彼らは，形式ニューロン，または人工ニューロンとよばれる素子を定義した．図 3.6 はその一例である．実際のニューロンのように，形式ニューロンも他の多数の形式ニューロン $i = 1, 2, \ldots$ から入力信号 x_i

を受け入れる．図 3.6 では入り込んでくる複数の矢印として入力を表している．ただし，ニューロンを出す信号はオンとオフの情報しかないとして，x_i の値は 0 か 1 しかとらないものとする．しかしシナプスごとに，ニューロン同士の結合の強さが違うため，重み w_i を導入して，層への総入力を

$$u = \sum_i w_i x_i \tag{3.71}$$

と定義する．u は細胞体に実際に入ってくる電気信号の総量に相当する．

総入力 u を受けて，ニューロンは「軸索」方向へ出力を出すが，そこには閾値があるはずである．その状況をヘビサイドの階段関数

$$\theta(x+b) = \begin{cases} 1 & (x \geq -b) \\ 0 & (x \leq -b) \end{cases} \tag{3.72}$$

でモデル化する．b は閾値を与えるパラメータである．すると，この形式ニューロンの出力は結局

$$z = \theta(u+b) = \theta\left(\sum_i w_i x_i + b\right) \tag{3.73}$$

となる．ここで用いた階段関数のように，総入力 u を出力 z へ変換する関数を一般的に活性化関数という．活性化関数に階段関数を用いため，z の値もまた，0 か 1 の 2 値となり，それを再び他のニューロンへの入力とすることができる．

マカロックとピッツの形式ニューロンを多数組み合わせることで，計算機と同じようにどんな論理演算でも実現することができる．任意の論理回路が NAND ゲートの組み合わせで実現できることは計算機科学でよく知られているため，この証明は NAND ゲートの出力を実現する形式ニューロンの回路を組めることを示せばよい．

NAND ゲートとは，2 つの 2 値入力 x_1, x_2 に対する出力値の関係が次のようになっている論理回路のことである．

(x_1, x_2)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
y	1	1	1	0

つまり，2 つの入力 x_1, x_2 からなる形式ニューロン (図 3.7) を考え，NAND ゲートによる出力 y を満たすような重みとバイアスの値を見つければよいということである．

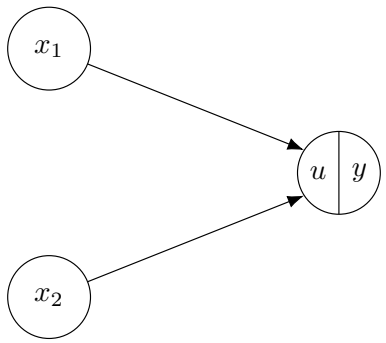


図 3.7: 簡単なニューロン回路の例

これは簡単に見つけることができ，例えば

$$\begin{aligned} y &= \theta(w_1 x_1 + w_2 x_2 + b) \\ &= \theta(-x_1 - x_2 + 1.5) \end{aligned} \tag{3.74}$$

とすればよい．実際

$$\begin{aligned} y(0,0) &= \theta(1.5) = 1 \\ y(1,0) &= \theta(0.5) = 1 \\ y(0,1) &= \theta(0.5) = 1 \\ y(1,1) &= \theta(-0.5) = 0 \end{aligned}$$

となり，NAND ゲートの出力そのものであると分かる．したがって，このニューロン回路を適切に組み合わせることにより，任意の論理演算を実現することができる．

3.4.3 パーセプトロン

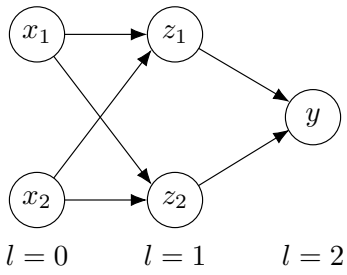


図 3.8: 2 層のパーセプトロン

形式ニューロンでどんな論理回路でも作ることがわかったが，解きたい問題をニューロン回路によって計算してくれる効率的な回路が設計できるがはわからない．そこでローゼンブラットは形式ニューロンのネットワークの重み w とバイアス b を解きたい問題に対してよく処理できるように訓練させるという「教師あり学習」のアイデアを付け加えた．ここでは，パーセプトロンの構造について説明する．

パーセプトロンとは形式ニューロンを層構造をなすように複数組み合わせて作ったニューロン回路のことである．図 3.8 はパーセプトロンの例で，$l = 0, 1, 2$ の 3 つの層で形成されている．最も左側にある層はデータが入力される部分であるため，入力層とよばれる．それに対して最も右側にある層は最終的な出力を行う層であることから，出力層とよばれる．そして，入力層と出力層の間にある層は中間層または隠れ層とよばれる．パーセプトロンでは層数に入力層を含めないため，これは 2 層のパーセプトロンである．

パーセプトロンで解きたい問題を解かせたい場合，各層で形式ニューロンの数をいくつに設定すればよいかというと，入力層は入力値のベクトル $\boldsymbol{x}^\top = (x_1, x_2, \cdots)$ の各成分 x_i を出力値として

もつ入力用の形式ニューロンの集まりであるため，入力ベクトルの次元の数に，出力層は推定したい最終結果 $\boldsymbol{t}^\top = (y_1, y_2, \cdots)$ の各成分 y_i の数で設定する必要がある．中間層の形式ニューロン数に関しては特に制限はない．さらに，中間層は層数を 2 つ以上に設定してもよい．

3.4.4 順伝播ニューラルネットワーク

これまでの形式ニューロンやパーセプトロンについて説明した．ここでは，それらをもとにした現代の順伝播ニューラルネットワークについて説明する．

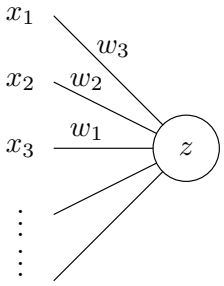


図 3.9: ユニットの構造．$x_1, x_2, \ldots$ の入力を受け，z を出力する

順伝播型ニューラルネットワークではユニットと呼ばれる素子からなる．図 3.9 はユニットの構造を示したものである．見た目はパーセプトロンにおける形式ニューロンと変わらない．しかし，ユニットは形式ニューロンとは違い，0,1 の値ではなく，実数値の入出力を持つことができる．また，もう 1 つの大きな違いは，活性化関数として階段関数ではなく，微分可能は増化関数を採用するということである．まず，ユニットにさまざまな入力 x_i は入ってきているとする．各ユニットによって結合強度が異なるため，実際にはそれらの重み付き和

$$u = \sum_i w_i x_i \tag{3.75}$$

がユニットへの総入力となる．u は活性とも呼ばれる．この活性にバイアス b を加えたうえで，さらに活性化関数 f での変換を施したものが，このユニットからの出力 z になる．

$$z = f(u + b) = f\left(\sum_i w_i x_i + b\right) \tag{3.76}$$

活性化関数は閾値 $-b$ を超えるとニューロンが信号を発することをモデル化する部分であるため，昔の研究では階段関数によく似た連続関数がよく用いられていた．その一例がシグモイド関数

$$\sigma(u + b) = \frac{1}{1 + e^{-u-b}} \tag{3.77}$$

である．図 3.10 をみると，閾値付近で信号が立ち上がっていることがわかる．

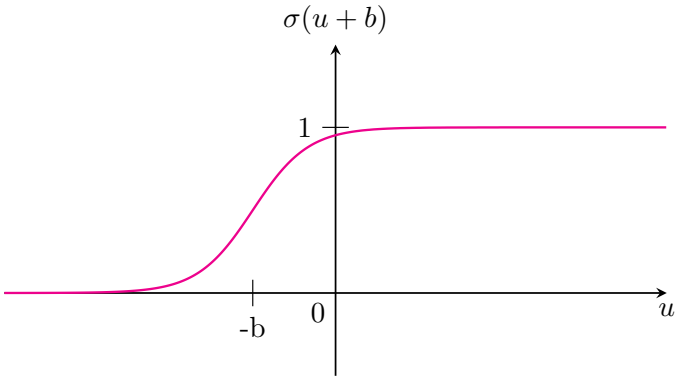


図 3.10: シグモイド関数

順伝播型ニューラルネットワークとはこのようなユニットを層状につなぎ合わせたアーキテクチャである．図 3.11 はその典型的な一例である[*4]．l は層数で，$l=0$ が入力層，$l=3$ が出力層，そして，$l=1,2$ が中間層を表している．信号の伝達は $l=0,1,2,3$ の順に進み，そのため順伝播型と呼ばれる．

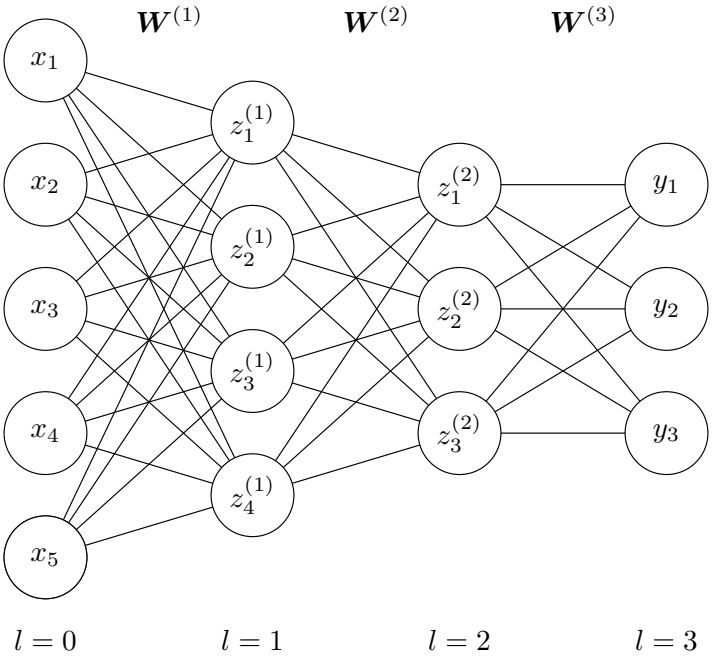


図 3.11: 入力層と 3 層からなる順伝播ニューラルネットワークの例．ユニットを表す丸の中に書かれた変数は，各ユニットの出力値を意味する．

一般的な順伝播型ニューラルネットワークを考える．まず，入力層の各ユニットはニューラルネットワーク全体への入力ベクトル $\boldsymbol{x}^\top=(x_1,x_2,\cdots)$ の各成分が入力して，それをそのまま出力

[*4] 本論文ではニューラルネットワークのユニット同士をつなぐ線を矢印ではなく，線で表すことにする．

する．つまり，入力ユニットは活性化関数をもたず，単に値 x_i を出力するだけである．入力層は入力ベクトル $\boldsymbol{x}$ の次元数だけユニットをもつ．また，「第 0 層からの出力である」という意味を込めて次のようなラベル付けをする．

$$\boldsymbol{z}^{(0)} = \boldsymbol{x} \tag{3.78}$$

次に中間層について考える．第 l 層中の j 番目のユニットを考える．このユニットは第 $l-1$ 層の各ユニットからの入力 $z_i^{(l-1)}$ を入力として受けるため．活性は

$$u_l^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)} \tag{3.79}$$

となる．ただし $w_{ji}^{(l)}$ は，第 $l-1$ 層のユニット i を第 l 層のユニット j の間の結合の重みである．

ユニットの出力は，活性 $u_j^{(l)}$ にユニット固有のバイアス $b_j^{(l)}$ を加え，さらに活性化関数 $f^{(l)}$ を作用させたもの

$$z_j^{(l)} = f^{(l)}\left(u_j^{(l)} + b_j^{(l)}\right) = f^{(l)}\left(\sum_i w_{ji}^{(l)} z_i^{(l-1)} + b_j^{(l)}\right) \tag{3.80}$$

で与えられる．[*5]

これらの式を行列表示しよう．まず l 層の全ユニットの活性をまとめてベクトル $\boldsymbol{u}^{(l)} = \left(u_j^{(l)}\right)$ で，出力をまとめて $\boldsymbol{z}^{(l)} = \left(z_j^{(l)}\right)$ で表す．また，第 $l-1$ 層のユニット i と第 l 層のユニット j の間の結合の重み $w_{ji}^{(l)}$ を (j,i) 成分とする重み行列

$$\boldsymbol{W}^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{11}^{(l)} & \cdots \\ w_{21}^{(l)} & w_{22}^{(l)} & \\ \vdots & & \ddots \end{pmatrix} \tag{3.81}$$

も導入する．さらにバイアスのまとめて縦ベクトル $\boldsymbol{b}^{(l)} = \left(b_j^{(l)}\right)$ で表記することにすると，中間層が行う演算処理は

$$\boldsymbol{u}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)}, \quad \boldsymbol{z}^{(l)} = f^{(l)}\left(\boldsymbol{u}^{(l)} + \boldsymbol{b}^{(l)}\right) \tag{3.82}$$

とまとめることができる．ただし，一般のベクトル $\boldsymbol{v}^\top = \begin{pmatrix} v_1 & v_2 & v_3 & \cdots \end{pmatrix}$ に対する関数 f の作用は，次のように各成分にそれぞれ作用するものとして定義することにする．

$$f(\boldsymbol{v}) = f\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} (f(\boldsymbol{v}))_1 \\ (f(\boldsymbol{v}))_2 \\ (f(\boldsymbol{v}))_3 \\ \vdots \end{pmatrix} \equiv \begin{pmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ \vdots \end{pmatrix} \tag{3.83}$$

これまで重みとバイアスを区別して書いたが，バイアスも重みに含めてしまうことができる．それを理解するためにすべての中間層 (と入力層) に，常に 1 という出力をするユニットを 1 つ加える．

[*5] 活性化関数は $f^{(l)}$ はユニットごとに変えても問題ないが，一般的には層ごとに共通の関数を用いる．

そしてそれを $i=0$ というラベルで呼ぶことにする．

$$z_0^{(l)} = 1 \tag{3.84}$$

第 $l-1$ 層の $i=0$ と第 l 層の $j \neq 0$ の間に重み w_{j0}^l を導入する．すると，これはバイアスそのものになる．なぜなら $w_{j0}^l = b_j^l$ とおくと，

$$\sum_{i=0} w_{ji}^l z_i^{l-1} = \sum_{i=1} w_{ji}^l z_i^{l-1} + w_{j0}^l = \sum_{i=1} w_{ji}^l z_i^{l-1} + b_j^l \tag{3.85}$$

となるからである．そのため，今後，必要のない場合は重みの中にバイアスも含めてしまい，バイアスをあらわに書かない次のような表記を用いることにする．

$$\boldsymbol{u}^{(l)} = \boldsymbol{W}^{(l)} \boldsymbol{z}^{(l-1)}, \quad \boldsymbol{z}^{(l)} = f^{(l)}\left(\boldsymbol{u}^{(l)}\right) \tag{3.86}$$

最後に出力層について考える．L 層からなる順伝播型ニューラルネットワークを考える．すると最後の第 L 層が出力層に相当する．出力層は，手前の第 $L-1$ 層から $\boldsymbol{z}^{(L-1)}$ を入力され，それを変換することでニューラルネットの最終出力 $\boldsymbol{y} = \boldsymbol{z}^L$ を出力する．出力層の役割は，回帰など機械学習で実行したいタスクを処理することである．つまり，$\boldsymbol{h} = \boldsymbol{z}^{(L-1)}$ が入力 $\boldsymbol{x}$ の表現であり，この表現 $\boldsymbol{h}$ の回帰分析を通じて $\boldsymbol{x}$ と $\boldsymbol{y}$ の関係を推定するのが出力層の役割である．

$$\hat{\boldsymbol{y}} = \boldsymbol{z}^{(L)} = f^{(L)}\left(\boldsymbol{u}^{(L)}\right), \quad \boldsymbol{u}^{(L)} = \boldsymbol{W}^{(L)} \boldsymbol{h} \tag{3.87}$$

出力層においては，活性化関数の値域が目標変数の値域と一致するように $f^{(L)}$ を選ぶ必要がある．

入力層から出力層に至る一連の情報処理の仕組みを説明した．結局のところ順伝播型ニューラルネットは，各総出の処理を順次入力ベクトル $\boldsymbol{x}$ に施していくことで，$\boldsymbol{x}$ から次の出力値を与える関数モデルということになる．

$$\hat{\boldsymbol{y}} = f^{(L)}\left(\boldsymbol{W}^{(L)} f^{(L-1)}\left(\boldsymbol{W}^{(L-1)} f^{(L-2)}\left(\cdots \boldsymbol{W}^{(2)} f^{(1)}(\boldsymbol{x})\right)\right)\right) \tag{3.88}$$

3.5 ニューラルネットワークによる機械あり学習

これまで，脳の神経回路を模した数理モデルとしてニューラルネットワークを定義した．ここでは，構築したニューラルネットワークに対して，機械学習によってタスクを遂行させるにはどうすればよいかを説明する．

順伝播ニューラルネットは入力 $\boldsymbol{x}$ を受け取ることで出力

$$\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{w}) \equiv \boldsymbol{y}\left(\boldsymbol{x}; \boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L)}, \boldsymbol{b}^{(1)}, \ldots, \boldsymbol{b}^{(L)}\right) \tag{3.89}$$

を出す．ただし，重み $\boldsymbol{W}^{(l)}$ とバイアス $\boldsymbol{b}^{(l)}$ をまとめて $\boldsymbol{w}$ と書いた．したがって，出力の値はパラメータ $\boldsymbol{w}$ によって決まることになる．この出力を機械学習における識別関数とみなすと，データを用いてパラメータ $\boldsymbol{w}$ に関してフィッティングすることができる．この手順は通常の教師あり

学習と同じであり，まず訓練データ $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1,\ldots,N}$ を用意する．そのうえで訓練データ $\boldsymbol{x}_n$ を入れた際のニューラルネットワークの出力値 $\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{w})$ と，目標値 $\boldsymbol{y}_n$ のズレができるだけ小さくなるように，重みとバイアスを調整して学習する．ズレを測る誤差関数 $E(\boldsymbol{w})$ の選び方は，考えるタスクと出力層の構造に依存する．そして誤差関数の最小化

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}}\ E(\boldsymbol{w}) \tag{3.90}$$

が学習に対応する

式 (3.89) のように考えると，入力から一気に $\boldsymbol{y}$ が得られているように見えるが，実際は層状の構造に従って信号が処理されていた．特に，第 $L-1$ 層までの情報処理と，最後の出力層を切り分けて考えてみよう．なぜなら，第 $L-1$ 層の部分までは，入力 $\boldsymbol{x}$ から層ごとに順次高次な表現を構成する役割を果たしているとみなせるからである．すると第 $L-1$ 層からの出力は，入力 $\boldsymbol{x}$ に対する深層表現 $\boldsymbol{h}$ である．

$$\boldsymbol{h}\left(\boldsymbol{x}; \boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L-1)}\right) = \boldsymbol{z}^{(L-1)}\left(\boldsymbol{x}; \boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L-1)}\right) \tag{3.91}$$

すると出力層は，この表現を使って回帰のような通常の機械学習を行う役割を担っているとみなせる．

では次に，ニューラルネットワークにさせたいタスクごとに分けて，出力層の構造を見ていく．

回帰の場合

もし表現 $\boldsymbol{h}$ について線形回帰を行いたいのであれば，自明な活性化関数，つまり単なる恒等写像を用いた出力層を用意すればよい．

$$\boldsymbol{y} = \boldsymbol{W}^{(L)}\boldsymbol{h} \tag{3.92}$$

このようなユニットは線形ユニットと呼ばれる．ただし多くの場合は線形ではない一般的な回帰を行いたいため，何らかの非自明な活性化関数を考える．

$$\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\theta}) = f^{(L)}\left(\boldsymbol{W}^{(L)}\boldsymbol{h}\right) \tag{3.93}$$

具体的にどのような $f^{(L)}$ を選ぶかは，問題の性質に応じて我々が設定する必要がある．

出力をデータでフィッティングするには，$\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\theta})$ を用いたモデルの予測値と実際のデータができるだけ近くなるように，平均二乗誤差を最小化する．

$$E(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{\theta}) - \boldsymbol{y}_n\right)^2, \quad \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ E(\boldsymbol{\theta}) \tag{3.94}$$

学習により得られた最適パラメータ $\boldsymbol{\theta}^*$ を代入したモデル $\boldsymbol{y}(\boldsymbol{x}; \boldsymbol{\theta}^*)$ は，入力 $\boldsymbol{x}$ の値に応じて $\boldsymbol{y}$ をよく予測できるようになるだろう．これが通常の回帰と異なるのは，回帰関数のパラメータだけでなく，表現を決める中間層の重みパラメータも同時に学習しているということである．これがニューラルネットワークによる表現学習である．

2 値分類の場合

回帰と並んで代表的なタスクは，データを 2 つのクラスに分類する 2 値分類であった．ただし，2 つのクラスに対応するラベル y の値は 0 か 1 の 2 値であるとする．ニューラルネットワークで 2 値分類を表現するには，出力層が表現 $\boldsymbol{h} = \boldsymbol{z}^{(L-1)}$ のロジスティック回帰を与えるようにデザインすればよい．

訓練データ $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1,\ldots,N}$ の標準値 y_n は 0 か 1 だが，ロジスティック回帰では，この 2 値変数そのものではなく，y が 1 である確率 $\hat{y} = P(y = 1|\boldsymbol{x})$ を推定する．言い換えれば y の期待値を推定する．なぜなら，2 値変数 y の従うベルヌーイ分布のもとで期待値は $E_{P(y|\boldsymbol{x})}[\mathrm{y}|\boldsymbol{x}] = \sum_{y=0,1} yP(y|\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$ となるからである．したがってロジスティック回帰を行うには，出力層のユニットが 1 つでその出力値が $P(y = 1|\boldsymbol{x})$ を推定するようなニューラルネットワークを考えるのが自然である．つまり，出力層の構造は式 (**??**) に合わせて，次のようにすればよい．

$$y(\boldsymbol{x};\boldsymbol{\theta}) = P(y = 1|\boldsymbol{x};\boldsymbol{\theta}) = \sigma\left(\sum_i w_i^{(L)} h_i\right) \tag{3.95}$$

このようなユニットはシグモイドユニットと呼ばれる．シグモイドユニットの活性化関数はまさにロジスティック回帰におけるシグモイド関数であり，ユニットへの総入力が対数オッズである．

$$f^{(L)} = \sigma, \quad u^{(L)}(\boldsymbol{x};\boldsymbol{\theta}) = \ln \frac{P(y = 1|\boldsymbol{x};\boldsymbol{\theta})}{1 - P(y = 1|\boldsymbol{x};\boldsymbol{\theta})} \tag{3.96}$$

このニューラルネットワークが訓練できれば，学習済みモデルに分析したいデータ $\boldsymbol{x}$ を入力した際の出力値から，$\boldsymbol{x}$ のどちらのクラスに属するかを判定できる．なぜならば，$P(y = 1|\boldsymbol{x})$ が 1/2 を越えれば $y = 1$ のクラス，下回れば $y = 0$ のクラスに属するのが尤もらしいと判断できるからである．

ではこのようなニューラルネットワークはどのように学習させればよいか．出力層はロジスティック回帰の構造をもっているため，ロジスティック回帰同様，最尤法を用いればよいことがわかる．つまり，ニューラルネットワークが $P(y = 1|\boldsymbol{x};\boldsymbol{\theta})$ を推定するベルヌーイ分布

$$P(y|\boldsymbol{x};\boldsymbol{\theta}) = P(y = 1|\boldsymbol{x};\boldsymbol{\theta})^y (1 - P(y = 1|\boldsymbol{x};\boldsymbol{\theta}))^{(1-y)} \tag{3.97}$$

を推定することと同じであるため，この分布の負の対数尤度を誤差関数にすればよい．したがって出力 $y(\boldsymbol{x}_n;\boldsymbol{\theta}) = P(y = 1|\boldsymbol{x}_n;\boldsymbol{\theta})$ に対し，

$$E(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \left(y_n \ln y(\boldsymbol{x}_n;\boldsymbol{\theta}) + (1 - y_n) \ln 1 - y(\boldsymbol{x}_n;\boldsymbol{\theta})\right) \tag{3.98}$$

が誤差関数であり，これを最小化するパラメータを見つけることが学習である．

このようにニューラルネットワークによって，ロジスティック回帰に向いた深層表現を表現学習できることになる．

多クラス分類の場合

最後に多クラス分類について考える．例えば手書き文字認識や画像認識は典型的な多クラス分類である．いまクラスが K 個だけあるとする．K が 3 以上の場合は，表現 $\boldsymbol{h} = \boldsymbol{z}^{(L-1)}$ をソフトマックス回帰する出力層を作ればよい．

各クラスに対して目標変数が $y = 1, 2, \ldots, K$ という値をとるものとする．そして出力層には K 個のユニットを用意し，それらの各ユニットは出力値として，入力 $\boldsymbol{x}$ が $y = k$ 番目のクラスに属する確率 $P(y = k|\boldsymbol{x})$ を推定するものとする．この出力層でソフトマックス回帰を表現するためには，出力層の k 番目のユニットは次の出力値をもてばよい．

$$y_k(\boldsymbol{x}; \boldsymbol{\theta}) = P(y = k|\boldsymbol{x}; \boldsymbol{\theta}) = \mathrm{softmax}_k \left(u_1^{(L)}, \ldots, u_K^{(L)} \right) \tag{3.99}$$

$$\boldsymbol{u}^{(L)} = \boldsymbol{W}^{(L)} \boldsymbol{h} \tag{3.100}$$

ここで出力層の活性化関数はソフトマックス関数である．学習済みニューラルネットワークへ分析したいデータ $\boldsymbol{x}$ を入力し，出力 y_k が最大になる k を所属クラスと判定する．

ソフトマックス関数には，目標変数をベクトル表示するのが便利である．

$$y \quad \longleftrightarrow \quad \boldsymbol{t}(y) = (t(y)_j) = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}^\top \tag{3.101}$$

右辺は $y = k$ のとき，第 k 成分のみが 1 で他の成分は 0 のベクトルである．訓練データも，この t にかんして与えることにする．

$$\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{t}_n)\}_{n=1,\ldots,N} \tag{3.102}$$

このベクトル $\boldsymbol{t}_n$ の第 k 成分を $t_{n,k}$ と書くと，ニューラルネットワークのソフトマックス出力に対する交差エントロピーは

$$E(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{n,k} \ln y_k(\boldsymbol{x}_n; \boldsymbol{\theta}) \tag{3.103}$$

であり，これが誤差関数に他ならない．

3.5.1 中間層で使われる活性化関数

ここでは，現在のニューラルネットワークの中間層でよく使われる活性化関数である双曲線正接関数と ReLU 関数を紹介する．

双曲線正接関数

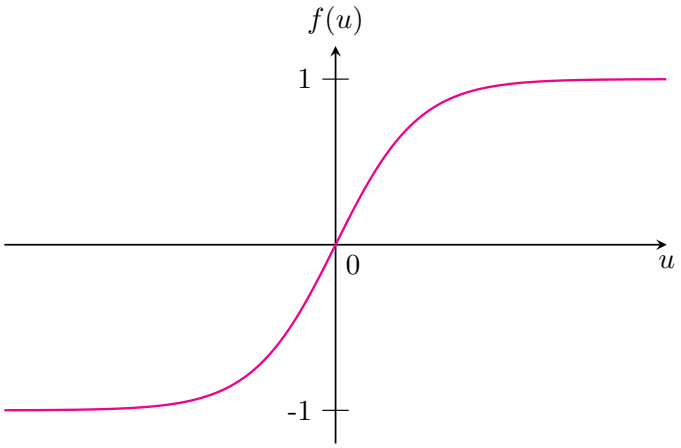


図 3.12: 双曲線正接関数

シグモイド関数 (3.10) の値域は $0 \leq \sigma(u) \leq 1$ であったが負の値も欲しい場合に双曲線正接関数が用いられる.

$$f(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} \tag{3.104}$$

ReLU 関数

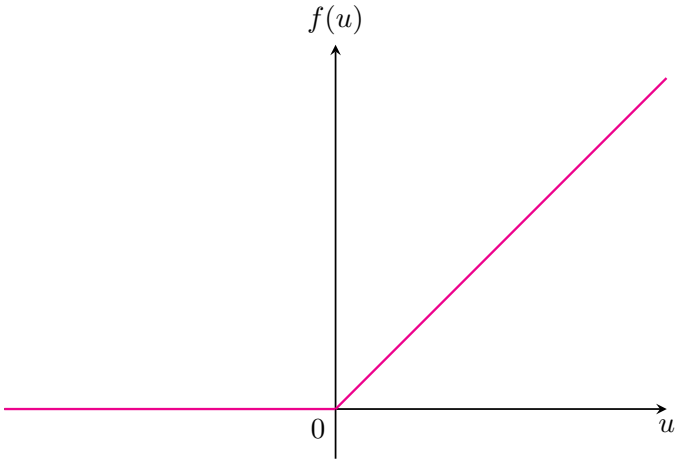


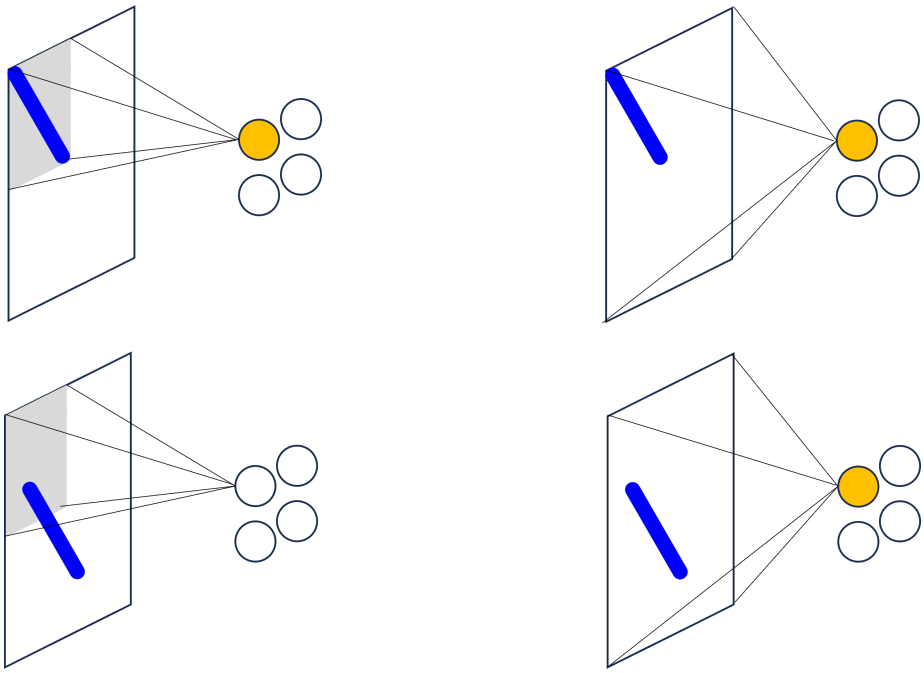
図 3.13: ReLU 接関数

シグモイド関数や双曲線正接関数を活性化関数として用いると，勾配消失という問題を引き起こすという大きな問題があった．この問題を解消できる活性関数が ReLU 関数である．

$$f(u) = \max\{0, u\} = \begin{cases} u & (x > 0) \\ 0 & (x \leq 0) \end{cases} \tag{3.105}$$

3.6 畳み込みニューラルネットワーク

3.6.1 一次視覚野と畳み込み



(a) 単純型細胞　　**(b)** 複雑型細胞

図 3.14: 単純型細胞と複雑型細胞

多くの目をもつ生き物には視界に入っている物体を認識し，それが何であるかを判断することができる．なぜそのようなことができるのかというと，視覚情報からパターンを認識する高い能力が備わっているからである．どのようなメカニズムでパターンを認識しているか，その詳細はまだわかっていないが，ニューロンのネットワークがもつ構造に理由があると考えられている．

1958 年にハーバード大学のヒューベルとウィーゼルは，猫の視覚野[*6]に特定の傾きをもつ線分を見せたときにだけ反応する細胞があることを発見した．さらにそのような細胞は 2 種類に大別できることを明らかにした．それらの細胞は単純型細胞と複雑型細胞とよばれ，受容野とよばれるニューロンを発火させるような入力を生じさせる領域の広さによって分別される．

[*6] 視覚野とは，視覚情報の処理に関連する脳の部位である．大脳皮質の後頭葉という部分に位置する．

図 3.14 は，網膜から視覚刺激を受け取ったときに，ニューロンがどう変化するかを示した概念図である．それぞれ左側ある四角が網膜の一部の領域に対応し，青で塗りつぶされた線分が視覚情報に対応する．そして，右側の 4 つの丸がニューロンであり，(a) は単純型細胞，(b) は複雑型細胞である．灰色で塗りつぶされた部分はそれぞれ左上のニューロンがもつ受容野で，見てわかるとおり単純型細胞における受容野は四角の一部の領域のみしか持たず，複雑型細胞は全領域を持っている．

単純型細胞 (a) に注目しよう．左上の図では線分が受容野にすっぽりと収まっており，ニューロンが発火している．しかし左下の図のように，線分の位置がずれると，受容野に収まらなくなり発火しなくなってしまう．それに対して，複雑型細胞 (b) は広い受容野をもつため，線分の位置がずれても発火し続けている．

2 種類の細胞により，ある特定の位置にあるパターンと，ある領域内にあるパターンの 2 種類のパターンを検出することが可能になる．

3.6.2 ニューラルネットワークと畳み込み

畳み込みニューラルネットワークは，主に 2 種類に層からできている．1 つ目は単純型細胞と類似のユニットからなる層で，局所的な受容野をもっている．パターン認識では，抽出すべきパターンが画像のどの位置に現れるかはあらかじめわからない．そこで図**??**(a) のように，入力側の層を，局所的な受容野をもつ単純型細胞で覆う．これにより，どの局所受容野にパターンが現れても次層のユニットに活性が生じる．これまで考えてきた全結合型のニューラルネットワークと比べ，構造が疎になっていることに注目してほしい．このように CNN では重み行列がスパースになっている．

CNN がパターン抽出のためのモデルであるならば，パターンが現れる画像中の位置によってパターン認識能力が変化しては困る．それを防ぐには，どこの局所受容野と単純型細胞の結合も，すべて同じ重みを共有すればよい．つまり図**??**(a) において同じ色の線で書かれた結合重みは，すべて同じパラメータを共有している．このようにすることで，学習中に訓練サンプルのある受容野に現れたパターンから学んだ結果を，ほかの位置に現れたパターンの抽出にも適用できるからである．したがって単純型細胞への入力はすべて同じ重み行列をもつため，結合の多さにもかかわらずパラメータの数は少数である．CNN においてこれを実現する層を畳み込み層と呼ぶ．

この一方，複雑型細胞は図**??**(b) のように，多数の単純型細胞の出力をとりまとめる役割を果たす．つまりパターン抽出には直接関与しない．そのためこの細胞へ入力する結合の重みは固定し，学習しないものとする．CNN においてこれを実現する層はプーリング層とよばれる．

3.6.3 画像データの扱い

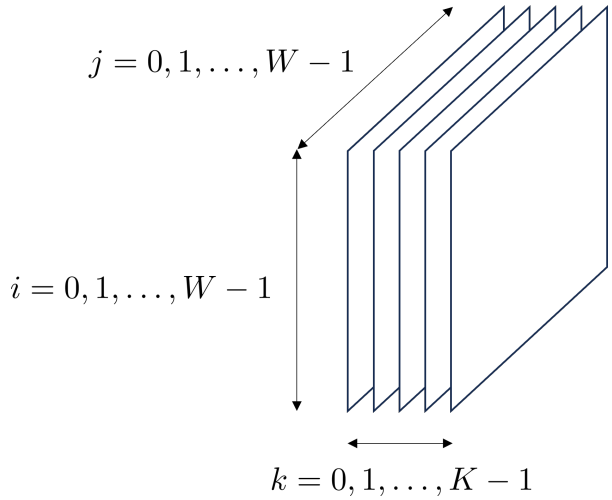


図 3.15: K チャンネルからなる $W \times W$ 画像

画像などの 2 次元データを用いて機械学習を行う場合，2 次元のデータをわざわざ 1 次元のベクトルに変換してモデルに入力するよりも，2 次元の構造を保ったままモデルに入力するほうが画像の 2 次元構造を最大限に活用できて便利である．ここでは，2 次元画像データをどう扱うかについて説明する．

モノクロ画像の場合，画像データは実数画素値 x_{ij} を持った 2 次元配列として表現することができる．添え字は画素の位置が (i, j) にあることを指定している．サイズが $W \times W$ の画像の場合，

$$i, j = 0, 1, \ldots, W-1 \tag{3.106}$$

の範囲で値をとる．0 から数え始めているのは数式をシンプルにするためである．一般的な画像の場合，画像の位置 (i, j) の自由度に加え，色成分などの自由度が付与される．このような自由度をチャンネルと呼ぶ．例として RGB カラーを用いた画像では位置 (i, j) に加え，赤，緑，青の 3 成分に対応した $k = 0, 1, 2$ の成分が加わり，画素値は x_{ijk} となる．一般化して K チャンネル $k = 0, 1, 2, \ldots, K$ を考えると図 3.15 のように，1 枚の画像も，各チャンネルに応じた K 枚の画像でできているとみなすことができる．

3.6.4 畳み込み層

まず 1 チャンネルの簡単な場合から考える．畳み込み層は行列入力 $(z_{ij}^{(l-1)})$ へフィルタを作用させる役割を果たす．そこでまずフィルタの役割を説明しよう．フィルタは入力よりも小さいサイズをもつ $H \times H$ 画像として定義する．その画素値を h_{pq} とする．画素の位置を表す添え字は

$$p, q = 0, 1, \ldots, H-1 \tag{3.107}$$

の値をとる．$(z_{ij}^{(l-1)})$ に対するフィルタ (h_{pq}) の畳み込みを次の演算て定義する．

$$u_{ij}^{(l)} = \sum_{p,q=0}^{H-1} z_{i+p,j+q}^{(l-1)} h_{pq} + b_{ij} \tag{3.108}$$

ここで，b_{ij} はバイアスである．この畳み込み演算を $*$ という記号で表すこともある．画像の画素 (i,j) にフィルタの画素 $(0,0)$ が重なるようして両者を重ねる．そしてフィルタと重なった位置での両者の画素値の積 $z_{i+p,j+q}^{(l-1)} h_{pq}$ を計算する．この値を重なりの領域全体にわたって足し合わせたものを $u_{ij}^{(l)}$ とする．これを畳み込み後の画像の (i,j) における画素値とする．この操作をフィルタが画像からはみ出ない範囲で行う．したがって畳み込める位置は $i,j = 0,1,\ldots,W-H$ で与えられ，畳み込みの画像サイズは $(W-H+1)\times(W=H+1)$ である．

フィルタは図**??**のように作用し，画像からあるパターンを抽出する．

このようにフィルタを畳み込むことで，画像から特定のパターンを抽出できる．CNN ではパターンを抽出するのに適したフィルタは，我々が与えるのではなく訓練データから学習させる．つまり，CNN におけるフィルタ h_{pq} が重みパラメータに他ならない．これらの重みは，かなり強く重み共有により正則化されている．というのもユニット (a,b) と次層のユニット (i,j) を結ぶ重みは，適当な整数 $\Delta_{1,2}$ で平行移動した場所のユニット $(a+\Delta_1, b+\Delta_2), (i+\Delta_1, j+\Delta_2)$ 間を結ぶ重みと全く同じだからである．つまり自由に動かせるパラメータは H^2 しかない．

次に K チャンネル画像の対する畳み込みを考える．K チャンネル画像は $z_{ijk}^{(l-1)}$ のように 3 つの添え字でラベルされているため，$W\times W\times K$ 画像とみなせる．このような画像を畳み込むには，同じチャンネル数を持つ $H\times H\times K$ フィルタ h_{pqk} を用意する．そのうえで，各チャンネルごとに先ほどの畳み込みを行い，その後同じ位置の画素は全チャンネルにわたって足し上げてしまう．その結果得られる次の $(W-K+1)\times(W-K+1)$ 画像が畳み込みの結果となる．

$$u_{ij}^{(l)} = \sum_{k=0}^{K-1}\sum_{p,q=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqk} + b_{ij} \tag{3.109}$$

多チャンネルのフィルタが 1 種類しかない場合，このように畳み込み後の画像は 1 種類しかない．畳み込み後も多チャンネル画像が欲しいならば，ほしいチャンネル数 M に応じて K チャンネルのフィルタを M 種類用意する．つまり

$$u_{ijk}^{(l)} = \sum_{k=0}^{K-1}\sum_{p,q=0}^{H-1} z_{i+p,j+q,k}^{(l-1)} h_{pqkm} + b_{ijm} \tag{3.110}$$

のように畳み込めばよい．

畳み込み層の最終的な出力は，この画像に活性化関数 f を作用させたものになる．

$$z_{ijm}^{(l)} = f(u_{ijm}^{(l)}) \tag{3.111}$$

畳み込み層からの出力のことを特徴量マップともいう．活性化関数としては，最近はよく ReLU 関数が用いられる．

このように $W \times W \times K$ 画像 $(Z_{ijk}^{(l-1)})$ から $(W-H+1) \times (W-H+1) \times M$ の画像 $(Z_{ijm}^{(l)})$ を出力するのが畳み込み層である．とはいえ，式 (**??**)，式 (**??**) からわかるように，この層は $w \times W \times K$ 個のユニットからなる $l-1$ 層と $(W-H+1) \times (W-H+1) \times M$ 個のユニットの l 層を，共有させた重み h_{pqkm} で結合させた普通の順伝播層ともみなせる．したがって通常の誤差逆伝播法を用いることで，重みであるフィルタを学習することができる．

ストライド

これまでフィルタは 1 目盛りずつ動かして，パターン抽出を行っていた．しかしもう少し大雑把に画像の特徴を捉えたい場合には，必ずしも 1 画素ずつ動かして一のパターンをこまめに調べる必要はない．そこで CNN ではストライドという方法がとられる．ストライド S とは，画素を $S-1$ 個余分に飛ばしてフィルタを動かしていくことを意味する．フィルタが S 画素動くごとに 1 回畳み込みを行うストライド S では，畳み込みの式は次のようになる．

$$u_{ijk}^{(l)} = \sum_{k=0}^{K-1} \sum_{p,q=0}^{H-1} z_{Si+p,Sj+q,k}^{(l-1)} h_{pqkm} + b_{ijm} \tag{3.112}$$

このようなストライドを用いた場合の畳み込み後の画像のサイズは

$$\left(\left\lfloor \frac{W-H}{S} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{W-H}{S} \right\rfloor + 1 \right) \tag{3.113}$$

となる．ここで床記号 $\lfloor x \rfloor$ は x を超えない最大の整数を意味する．このようにストライドを用いることで畳み込み後の画像サイズを小さくすることができる．しかしあまり大きなストライドを用いるとこまめな構造を捉えられず，捉えるべきパターンをストライドのせいでスキップしてしまう可能性もある．したがって考えるデータの性質に応じて，ストライドの設定には注意を払う必要がある．

パディング

畳み込みをすると必ず画像サイズが小さくなってしまう．特にフィルタサイズやストライドが大きいときはその縮小が顕著である．これはフィルタを重ね合わせる際に，元の画像からはみ出せないという制約に起因している．しかし，画像処理では，画像サイズが小さくなると技術的に都合がよくない状況も存在する．

そのような状況の場合，パディングという手法を用いることで，畳み込み後の画像サイズをある程度大きくすることができる．パディング P とは，元の画像の周りに厚さ P のふちを加える操作である．したがってストライド S，パディング P の畳み込みをした後の画像サイズは

$$\left(\left\lfloor \frac{W-H+2P}{S} \right\rfloor + 1 \right) \times \left(\left\lfloor \frac{W-H+2P}{S} \right\rfloor + 1 \right) \tag{3.114}$$

となる．パディングには加えた画素にどのような値を入れるかによって複数種類が存在する．よく使われるのがゼロパディングで，増やした画素すべてに画素値 0 を代入する．

ゼロパディングはとても簡単な手法であるが，0 を用いるためどうしても畳み込み後の画像は周辺部の画素値が小さくなり，暗くなってしまう．そこで，ゼロパディング以外にも，元の画像の画素値を用いてパディングする方法がある．名前だけ紹介すると周期パディングや反射パディングである．

3.6.5 プーリング層

次に複雑型細胞に対するプーリング層について説明する．この層の役割は，畳み込み層で捉えた局所的なパターンを，その位置がある程度移動しても捉えられるようにすることであった．入力位置の平行移動に対してロバストにするためには，各位置で局所的パターンを抽出して単純型細胞からの出力を合算すればよかった．

プーリングではまず，$W \times W \times K$ 入力画像 $z_{ijk}^{(l-1)}$ の各チャンネルに対し，各画素位置 (i, j) を中心とした $H \times H$ の大きさの領域 $\mathcal{P}_{ij}$ を考える．ただし画像はパディングしておき，どの点に対しても $\mathcal{P}_{ij}$ の領域を考えられるようにしておく．そして $\mathcal{P}_{ij}$ 中の画素値から，その領域での代表的な画素 $u_{ij}^{(l)}$ を決定する．代表値を決める方法は複数あるが最大プーリングでは，領域内の最も大きな画素値を代表値として取り出す．

$$u_{ijk}^{(l)} = \max_{(p,q)\in\mathcal{P}_{ij}} z_{pqk}^{(l-1)} \tag{3.115}$$

また，最大値の代わりに，領域内の平均を用いる平均プーリングという方法もよく使われる．

$$u_{ijk}^{(l)} = \frac{1}{H^2} \sum_{(p,q)\in\mathcal{P}_{ij}} z_{pqk}^{(l-1)} \tag{3.116}$$

なおプーリング層の役割は畳み込み層の出力を束ねるだけであるため，重みの学習は行わない．したがって学習中もプーリングの式は一切変更されない．

3.7 勾配降下法

ニューラルネットワークの学習は損失関数の最小化によって行われる．つまり，ネットワークパラメータの空間上でスカラー関数 $L(\boldsymbol{\theta})$ の最小値を探すという最適化問題を解くことになる．

$$\boldsymbol{\theta}^* = \operatorname*{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \tag{3.117}$$

損失関数は基本的に多くのパラメータを含む複雑な関数であるため，このような最適化問題を厳密に解くことはできない．そのため，計算機によって数値的に近似解を求めることになる．

3.7.1 勾配降下法

誤差関数 $E(\boldsymbol{\theta})$ の最小点を見つける手法のうち，最も直観的で単純なアイデアは図**??**のように，誤差関数のグラフの形状をした凹みにおいて．上のほうからボールを転がり落してボールの一番低いところにたどり着くまで待つ方法である．これがまさに勾配降下法の考え方である．

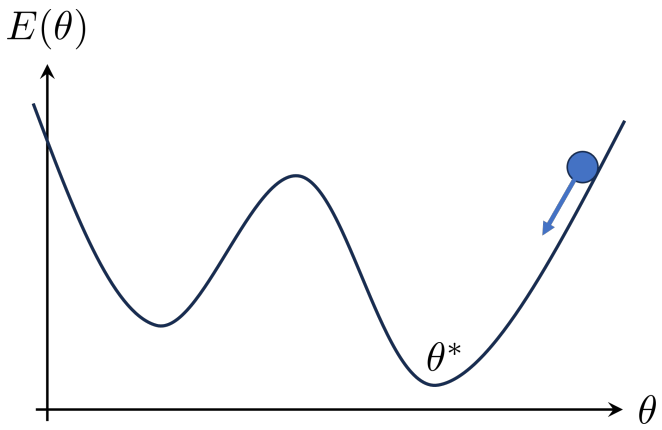


図 3.16: 勾配降下法による極小値の見つけ方

ボールを転がり始めさせる位置に対応して，勾配降下法ではパラメータの初期値 $\theta^{(0)}$ を用意する．この初期値から始めて，坂道でボールを転がすような操作を，離散的な時間 $t = 0, 1, 2, \ldots$ を用いて定式化しよう．坂を転がすというのは，現在位置におけるグラフの勾配

$$\nabla E(\boldsymbol{\theta}) = \frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \equiv \left(\frac{\partial E(\boldsymbol{\theta})}{\partial \theta_1}, \cdots, \frac{\partial E(\boldsymbol{\theta})}{\partial \theta_D} \right) \tag{3.118}$$

の逆方向に動かすことを意味する．ここで表記を簡単にするため，ニューラルネットワークの重みパラメータを下付き添え字 $\theta_1, \theta_2, \ldots, \theta_D$ でラベル付けした．D はニューラルネットワークの全パラメータ数である．すると時刻 t で位置 $\boldsymbol{\theta^{(t)}}$ にあったボールを勾配の逆方向に動かす際のルールは次のようになる．

$$\boldsymbol{\theta^{(t+1)}} = \boldsymbol{\theta^{(t)}} + \Delta\boldsymbol{\theta^{(t)}}, \quad \Delta\boldsymbol{\theta^{(t)}} = -\eta \nabla E\left(\boldsymbol{\theta}^{(t)}\right) \tag{3.119}$$

右辺で次時刻の $\theta^{(t+1)}$ を定義する操作を $t = 0, 1, 2, \ldots$ と順次繰り返していくことで，どんどんと誤差関数のグラフの底のほうに降りていくことができる．1 ステップでの移動距離 $\Delta\boldsymbol{\theta^{(t)}}$ の大きさを決めるハイパーパラメータ η は学習率 (learning late) と呼ばれる．この操作が収束し，もはや動けなくなった点が勾配が消える極小値である．ここでいう収束とは，計算機の数値制度の範囲内でもはや $\theta^{(t)}$ の変化が見られなくなる状況のことである．

学習率の取り方には一般論は存在せず，現状ではトライアルアンドエラーに基づかざるを得ない．しかし学習率をあまり大きくすると，1 ステップの刻みが荒らすぎて誤差関数の形状をうまく捉えられずに，収束に問題を起こす．その一方であまり小さくしすぎると学習が一向に進まなくなる．したがって，程よい大きさの学習率を見つけることが，学習をうまく進めるために重要となる．

3.7.2 局所的最小値の問題

いままでは最小値と極小値の区別に注意を払わなかった．誤差関数が下に凸な関数である簡単な状況では任意の極小値は必ず最小値に一致するので区別する必要がない．しかしながらディープ

ラーニングにおける誤差関数は一般的にとても複雑な非凸関数なので，本当の極小値である大域的極小値以外にも，膨大な数の局所的極小値をもつ．

さらにニューラルネットワークには高い対称性と極小値の重複がある．例えば，第 l 層の 2 つのユニット $j = 1, 2$ に注目すると，この 2 つのユニットを入れ替えても最終層の出力は変わらない．なぜなら，$\theta_{1i}^{(l)}, \theta_{k1}^{(l+1)}$ と $\theta_{2i}^{(l)}, \theta_{k2}^{(l+1)}$ をすべて同時に入れ替えれば何もしていないことと同じになるからである．各層でこのような入れ替えは ${}_{d_l}C_2$ 通りだけある．したがってニューラルネットワーク全体では $\prod_l {}_{d_l}C_2$ 通りだけの入れ替え対称性があることがわかる．つまり，局所的極小値が 1 つでも見つかれば，自動的に $\prod_l {}_{d_l}C_2$ 個の極小値が重複して存在することになる．このように，深層モデルでは極小値の数は膨大になる．

このような場合，勾配降下法で大域的極小値を探すのは，干草の中から針を探すようなものである．したがってディープラーニングでは極小値にはたどり着けても，真の極小値にはまずたどり着けない．通常の機械学習の文脈ではこれは深刻な問題であり，局所的最小値の問題や局所的最適解の問題と呼ばれる．

ところが不思議なことに，ディープラーニングでは真の最小値を見つけずとも，誤差関数のよい極小値さえ見つけられれば十分であると予想されている．これはディープラーニングを他の機械学習とは一線を画す画期的な手法にしていると同時に，ディープラーニングにおける大きな謎の 1 つである．この点に関しては現在でもさまざまな研究がなされている．

3.7.3 確率的勾配降下法

ディープラーニングが真の最小値を必要としないとはいっても，誤差関数の値があまりに大きい臨界点にはまり込んでしまっては全く使い物にならない．そこで臨界点にトラップされることをできるだけ回避するためにランダムな要素を取り入れて，はまり込んだ場所から弾き出す効果を生み出すことで勾配降下法を改良していく．

ランダムな要素を入れるためには，学習の仕組みを復習する必要がある．学習データ $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1,2,\ldots,N}$ が与えられたとき，誤差関数は各学習サンプル要素 $(\boldsymbol{x}_n, \boldsymbol{y}_n)$ で計算した誤差の和として表現できた．

$$E(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} E_n(\boldsymbol{\theta}) \tag{3.120}$$

例えば平均二乗誤差を用いるならば

$$E_n(\boldsymbol{\theta}) = \frac{1}{2} \left(\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{\theta}) - \boldsymbol{y}_n \right)^2 \tag{3.121}$$

であり，K クラス分類ならば交差エントロピー

$$E_n(\boldsymbol{\theta}) = -\sum_{k=1}^{K} t_{nk} \log y_k(\boldsymbol{x}_n; \boldsymbol{\theta}) \tag{3.122}$$

を用いた．先ほどの勾配降下法では式 (3.120) のように毎回の更新ですべての訓練サンプルを用いていた．このような方法はバッチ学習と呼ばれる．

しかし勾配によるパラメータ更新は何回も繰り返すことになるので，1 回の更新で毎回すべてのサンプルを用いる必要がない．各時間ステップで，一部の訓練サンプルだけを用いる方法をミニバッチ学習という．ミニバッチ学習ではまず，各時間 t で用いる訓練サンプルの部分集合 $\mathcal{B}^{(t)}$ を用意する．この $\mathcal{B}^{(t)}$ のことをミニバッチと呼び，通常は学習前にランダムに作成しておく，そして時刻 t における更新ではミニバッチ上で平均した誤差関数

$$E^{(t)}(\boldsymbol{\theta}) = \frac{1}{\mathcal{B}^{(t)}} \sum_{n \in \mathcal{B}^{(t)}} E_n(\boldsymbol{\theta}) \tag{3.123}$$

を用いる．ここで $n \in \mathcal{B}^{(t)}$ はミニバッチに含まれる訓練サンプルのラベルを表し，$|\mathcal{B}^{(t)}|$ はミニバッチの中のサンプル要素の総数である．これを用いてバッチ学習同様，パラメータを更新する．

$$\boldsymbol{\theta^{(t+1)}} = \boldsymbol{\theta^{(t)}} + \Delta\boldsymbol{\theta^{(t)}}, \quad \Delta\boldsymbol{\theta^{(t)}} = -\varepsilon \nabla E^{(t)} \left(\boldsymbol{\theta}^{(t)} \right) \tag{3.124}$$

特に各時刻のミニバッチに 1 つの訓練サンプルしか含まない $|\mathcal{B}^{(t)}| = 1$ という場合をオンライン学習や確率的勾配降下法と呼ぶ．

ミニバッチ学習ではランダムにミニバッチを選んだことにより，時刻ごとに誤差関数 $E^{(t)}(\boldsymbol{\theta})$ の形もランダムに変化する．したがって，ずっと同じ関数 $E(\boldsymbol{\theta})$ を使い続けるバッチ学習とは違い，望ましくない臨界点にはまり込む可能性がかなり小さくなる．これがミニバッチ学習が重宝される大きな理由である．

さらにサンプルを効率的に使う観点からもミニバッチは好まれる．訓練データのサイズが大きくなると，似たようなサンプルが含まれる可能性が高くなる．そこで訓練データ集合全体ではなく，ミニバッチを使用することで，1 回の更新ステップにおいて似たデータを重複して使用する無駄が省けるようになる．

またミニバッチでの勾配降下法では，各勾配 $\nabla E_n(\boldsymbol{\theta})$ の計算が独立で容易に並列化できる．したがって，コアのたくさんある GPGPU などの並列計算環境がある場合には，ある程度のサイズのミニバッチを利用するほうが理にかなっている．

3.7.4 ミニバッチの作り方

(ミニ) バッチ学習では，学習時間をエポック (epoch) という単位ごとに分けて考える．1 エポックという単位は，訓練データ全体を 1 周すべて使い切る時間単位を意味する．

まずはじめのエポックでは，データを適当なサイズのミニバッチにランダムに分割する．そして，これらのミニバッチを用いて勾配を更新していく．すべてのミニバッチを使い切ったら，このエポックは終了になる．しかし通常は，1 エポックでは不十分で，次のエポックに進み，改めてランダムにミニバッチを作り，同じプロセスを繰り返していく，そして，誤差関数が十分小さくなるまでエポックを繰り返したら，学習終了になる．

また，バッチ学習では毎回データを使い切るため，エポックと更新時間は一致する．

3.8 改良された勾配降下法

3.8.1 勾配降下法の課題

前節で勾配法には，局所最適解の問題があることをみたが，それ以外にも多くの課題がある．

図 3.17(a) のように誤差関数が深い谷を作っている状況を考える．このような谷に対して勾配降下法で勾配の方向にパラメータを更新するだけでは谷に沿って激しく振動するばかりで，いつまでたってもパラメータが収束しない．

一方，図 3.17(b) のように，平らな領域があった場合を考える．このような場所はプラトーと呼ばれる．一度プラトーに入り込んでしまうと勾配が消えるため，パラメータの更新も止まってしまう．ミニバッチアドでランダムな要素を入れたとしても，深層学習の誤差関数に現れるプラトーは学習の進みを遅くする原因になってしまう．

さらに，図 3.17(c) のような急に切り立った絶壁の存在も危険である．それまで緩やかな坂を下ってきたとしても，絶壁に当たった瞬間に極めて大きな勾配によって吹き飛ばされてしまう．

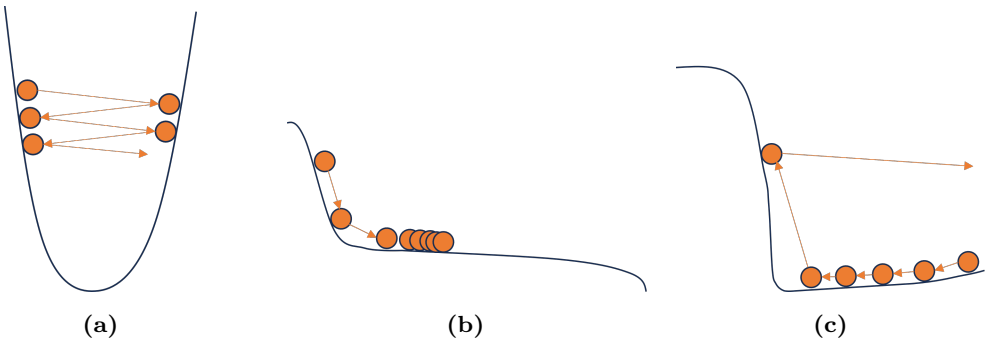


図 3.17: (a) 谷で振動.(b) プラトーで停止．(c) 絶壁で反射

3.8.2 モーメンタム法

振動による問題を改善する手法としてモーメンタム (momentum) というものがある．モーメンタムは「慣性」とも訳され．前時刻での勾配の影響を引きずらせることで振動を防ぐことができる．

振動の原因は，深い谷の底周りで急激な勾配が正負交互に発生するためであった．そこで，1 個前のステップでの勾配の影響を，現在の勾配に加えることを考える．前回のパラメータ更新量 $\Delta\boldsymbol{\theta}^{(t-1)}$ が負の値で，現在の勾配 $\nabla E(\boldsymbol{\theta}^{(t)})$ が正の大きな値であるとする．すると前回の更新量を現在の勾配に少し加えることで，図 3.18 のように今回の更新量 $\Delta\boldsymbol{\theta}^{(t)}$ を比較的小さな値に抑えることができる．つまり，手法で勾配が大きく正負に振れること防ぐことができる．

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)} \tag{3.125}$$

$$\Delta\boldsymbol{\theta}^{(t)} = \mu\Delta\boldsymbol{\theta}^{(t-1)} - (1-\mu)\eta\nabla E(\boldsymbol{\theta}^{(t)}) \tag{3.126}$$

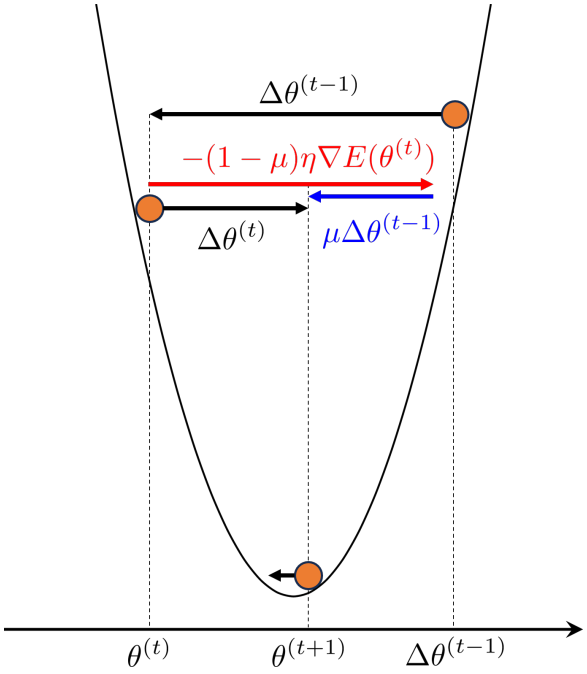


図 3.18: モーメンタム法による谷での振動の抑制

初期値は $\Delta\boldsymbol{\theta}^{(0)} = 0$ にとる．また，μ は比較的 1 に近い 0.5 ～ 0.99 程度の値をとる．

モーメンタム法は振動を防ぐ効果だけではなく，普通の斜面において勾配法でのパラメータ更新の加速も可能にする．これを見るために誤差関数の傾き $\nabla E(\boldsymbol{\theta}^{(t)})$ が一定の領域を考える．この場合，式 (**??**) の終端速度は $\Delta\boldsymbol{\theta}^{(t)} = \Delta\boldsymbol{\theta}^{(t+1)} = \Delta\boldsymbol{\theta}$ として更新式を解くことで，

$$\Delta\boldsymbol{\theta} = -\eta\nabla E(\boldsymbol{\theta}) \tag{3.127}$$

となる．つまり傾きの一定の斜面では，もともとの式 (**??**) においては $(1-\mu)\eta$ であった学習率が η にまで増える加速効果を与える．また，$\mu = 0$ にとるとモーメンタムの効果は消え，通常の勾配法に帰着する．

3.8.3 AdaGrad

モーメンタム法では勾配法の振動を防いだり，パラメータ更新を加速させたりする効果があった．ここでは，異なる側面に注目した改善策を説明する．

勾配降下法にはこれまで 1 つの学習率 η しかなかった．しかし実際には，パラメータ空間の座標方向によって誤差関数の勾配に大きな違いが現れることが想定できる．例えば，w_1 方向には急激な勾配を持つ一方，w_2 方向には緩やかな傾きしか持たない誤差関数があったとする．すると w_1 方向には大きな勾配に従い大きくパラメータ値が更新される一方，w_2 方向には一向にパラメータ更新が進まない．これは勾配法をコントロールする学習率が 1 つしかないからである．

もし各パラメータ方向に応じて複数の学習率が導入できれば，どの方向にも均等な速度で学習を

進めることができ，勾配降下法の収束性が良くなるはずである．ただし，むやみにやたらに学習率を増やしてしまっては，それらを我々がトライアンドエラーによって決めなくてはいけなくなり都合がよくない．そこでここでは，パラメータをあまり増やさずに，各方向に適切な有効学習率を持たせる手法を説明する．

このような手法のうち，早くから用いられていたものが AdaGrad(adaptive subgradient descent) である．

$$\Delta\boldsymbol{\theta}_i^{(t)} = -\frac{\eta}{\sqrt{\sum_{s=1}^{t}\left(\nabla E(\boldsymbol{\theta}^{(s)})_i\right)^2}}\nabla E(\boldsymbol{\theta}^{(t)}) \tag{3.128}$$

左辺は $\Delta\boldsymbol{\theta}^{(t)}$ の第 i 成分であり，$\nabla E(\boldsymbol{\theta}^{(t)})_i$ は勾配の第 i 成分である．AdaGrad では，過去の勾配成分の二乗和の平方根で学習率を割っている．それにより，すでに大きな勾配値をとってきた方向に対しては勾配を小さく減少させ，いままで勾配の小さかった方向への学習率を増大させる効果がある．これにより，学習が一向に進まない方向が生じてしまうことを防ぐことができる

AdaGrad の欠点をしては，学習の初期に勾配が大きいとすぐさま更新量 $\Delta\boldsymbol{\theta}_i^{(t)}$ が小さくなってしまうことである．そのままだと学習がストップするので，適切な程度にまで η を大きく選んであげる必要がある．そのために AdaGrad は学習率の選び方に敏感で使いにくい方法である．また，はじめから勾配が大きすぎるとすぐさま更新が進まなくなるため，重みの初期値にも敏感である．

3.8.4 RMSprop

RMSprop はヒントンにより講義の中で紹介された手法である．論文として出版していないにも関わらず，世界中で広く用いられている手法として有名である．

AdaGrad の問題は，ひとたび更新が 0 になってしまったら二度と有限の値に戻らないことであった．これは過去のすべての勾配の情報を収集してしまっていたためである．そこで十分過去の勾配の情報を指数的な減衰因子によって消滅させるように，二乗和ではなく指数的な移動平均 $v_{i,t}$ から決まる root mean square(RMS) を用いることにする．

$$v_{i,t} = \rho v_{i,t-1} + (1-\rho)(\nabla E(\boldsymbol{\theta^{(t)}})_i)^2 \tag{3.129}$$

$$\Delta\boldsymbol{\theta}_i^{(t)} = -\frac{\eta}{\sqrt{v_{i,t}+\epsilon}}\nabla E(\boldsymbol{\theta^{(t)}})_i \tag{3.130}$$

初期値は $v_{i,0} = 0$ とする．また，ϵ は分母が 0 とならないように導入した．よく $\epsilon = 10^{-6}$ の値が用いられる．また簡単のために

$$\mathrm{RMS}[\nabla E_i]_t = \sqrt{v_{i,t}+\epsilon} \tag{3.131}$$

という記法を今後用いる．

RMSprop では最近の勾配の履歴のみが影響するため，更新量が完全に消えてしまうことはない．また，深層学習では，うまく鞍点を抜け出した阿多は更新を加速したいため，モーメンタム法などと組み合わせて用いられる．RMSprop の有効性は広く実証されている．

3.8.5 AdaDelta

RMSprop は AdaGrad を改善したとても良い方法である．しかしながら，全体の学習率 η の値に敏感であるという事実にはあまり改善が見られない．その理由の 1 つは，次元性のミスマッチである．物理における次元解析を思い出そう．いま誤差関数 E は無次元であるとする．つまりパラメータ $\boldsymbol{\theta}$ を測るスケールを何倍しても不変な関数とする．その一方 $\Delta\boldsymbol{\theta}$ は長さの次元を持っており，E の微分は長さの逆数の次元を持つ．

$$[\Delta\boldsymbol{\theta}] = \mathrm{L}, \quad [\nabla E(\boldsymbol{\theta})] = \mathrm{L}^{-1} \tag{3.132}$$

3.8.6 Adam

Adam では，式 (**??**) の分母にある勾配の RMS のみならず，勾配自身も指数的な移動平均による推定値で置き換える．これは RMSprop の勾配部分に，指数的減衰を含むモーメンタムを適用したようなものであるが．実際は Adam はもっと手が込んでいる

まず勾配とその 2 乗の指数的な移動平均を定義する．

$$m_{i,t} = \rho_1 m_{i,t-1} + (1-\rho_1)\nabla E(\boldsymbol{w^{(t)}})_i v_{i,t} = \rho_2 v_{i,t-1} + (1-\rho_2)\left(\nabla E(\boldsymbol{w^{(t)}})_i\right)^2 \tag{3.133}$$

ただし初期値は $m_{i,0} = v_{i,0} = 1$ である．これは一見すると勾配の 1 次モーメントと 2 次モーメントのよい推定量のように見えるが，実はバイアスをもっている．というのも初期値は 0 にとってしまうので，更新の初期はモーメントの推定量が 0 のほうに偏ってしまう．

そこでバイアスを補正し，できるだけ不偏推定量に近づくようにする．例として 2 次モーメント $v_{i,t}$ を考える．初期値のもとで式 (**??**) を解くと

$$v_{i,t} = (1-\rho_2)\sum_{s=1}^{t}(\rho)^{t-s}\left(\nabla E(\boldsymbol{w})^{(s)_i}\right)^2 \tag{3.134}$$

である．SGD などでは各ステップで訓練サンプルがランダムにサンプリング去れるのに対応して，各時刻の勾配 $\nabla E(\boldsymbol{w}^{(s)})$ も，とある確率分布から毎時刻サンプリングされているものとみなせる．そこで式 (**??**) の期待値をとると

$$\begin{aligned}
\mathrm{E}[v_{i,t}] &= (1-\rho_2)\sum_{s=1}^{t}(\rho)^{t-s}\mathrm{E}\left[\left(\nabla E(\boldsymbol{w})^{(s)_i}\right)^2\right] \\
&\approx \mathrm{E}\left[(\nabla E(\boldsymbol{w^{(t)}})_i)^2\right](1-\rho_2)\sum_{s=2}^{t}(\rho_2)^{t-s} \\
&= \mathrm{E}\left[(\nabla E(\boldsymbol{w^{(t)}})_i)^2\right](1-(\rho_2)^t)
\end{aligned} \tag{3.135}$$

が得られる．2 行目の近似は，もし勾配が時間的に一定ならば厳密に成り立つ．そうでなくても減衰率を適切にとることで，勾配の値が大きく異なってしまうほどの十分に過去の寄与は小さく抑え

ることができるため，良い近似式であると思われる．したがって $v_{i,t}$ だけずれていることになる．これは t が小さいうちはとても大きなズレを生じさせる．そこでこの因子で割ることでバイアスを補正したモーメントの推定量

$$\hat{m}_{i,t} = \frac{m_{i,t}}{(1-(\rho_1)^t)}, \quad \hat{v}_{i,t} = \frac{v_{i,t}}{(1-(\rho_2)^t)} \tag{3.136}$$

を導入しよう．Adam はこれら推定値を用いた勾配降下法である

$$\Delta \boldsymbol{w}_i^{(t)} = -\eta \frac{\hat{m}_{i,t}}{\sqrt{\hat{v}_{i,t}+\epsilon}} \tag{3.137}$$

現論文におけるパラメータの推奨値は

$$\eta = 0.001, \quad \rho_1 = 0.9, \quad \rho_2 = 0.999, \quad \epsilon = 10^{-8} \tag{3.138}$$

である．さまざまな深層学習フレームワークでも，基本的にはこの推奨値が用いられている．

3.9 誤差逆伝播法

誤差逆伝播法とは多層ニューラルネットワークの学習手法である．この手法が 1986 年にアメリカの心理学者であったデビッド・ラメルハートらによって再発見されたことにより，第 2 次 AI ブームというのを巻き起こした．誤差逆伝播法の考えが発見されたのはこれよりも 20 年も前で，1967 年に甘利俊一によって多層ニューラルネットワークの勾配降下法の定式化された．なぜ，1980 年代の再発見でようやく誤差逆伝播法が注目されるようになったのか．それは計算機の発展によって誤差逆伝播法の実装が可能になり，その結果驚くべきパフォーマンスを発揮できることが世に知れ渡ったからである．

この節では，ニューラルネットワークの起源となったパーセプトロンの学習から入り，誤差逆伝播法の考え方と具体的な計算の仕組みを説明する．

3.9.1 パーセプトロンの学習測とデルタ則

ニューラルネットワークの学習を考える前に，3.4.3 章で紹介したニューラルネットワークの起源となった (古典) パーセプトロンの学習と，それを拡張した現代パーセプトロンの学習について考える．古典パーセプトロンとは脳の神経細胞であるニューロンをモデル化した人工ニューロンを層構造を持たせながらつなげたものであった．これは現代の視点でいれば，入出力の値が 0 か 1 の 2 値であるユニットを用いたニューラルネットワークと言える．

パーセプトロンを学習させ，望みの出力を生成させるにはどうすればよいかを理解するために，中間層なし，出力ユニット 1 つの簡単なパーセプトロンの学習について考える．パーセプトロンではユニット間を結ぶエッジにそれぞれ重み w_i が割り当てられている．この値は最初ランダムな値に初期化されているものとする．そして，パーセプトロンを学習させるために学習サンプル $\{\boldsymbol{x}, y\}$ を 1 つもってくる．パーセプトロンの入力層のユニットに学習データ $\boldsymbol{x}$ を入力すると，出力 $\hat{y}(\boldsymbol{x})$

が得られる．それを目標出力の値 y と比較し，もし値が異なっていれば正解の値が出力される方向に重みパラメータを調整する．これを何度も繰り返し行えば重みパラメータの最適化が進む．

パーセプトロンの場合，モデルの出力と目標出力が異なるのは 2 パターンのみである．まず，目標出力が $y = 1$ であるのに，実際の出力が $\hat{y}(\boldsymbol{x}) = 0$ となった場合である．この場合は出力ユニットへの入力 $u = \sum_i w_i x_i$ の値が小さすぎるということになる．したがって，入力ユニットから多くの信号が入ってくるように，発火している入力ユニット $x_i = 1$ との結合を強めてやればよいことになる．そこで発火しているユニット x_i との結合定数 w_i をそれぞれ

$$w_i \to w_i + \eta x_i \tag{3.139}$$

と値を更新する．ここで，η は学習率である．もう一つは目標出力が $y = 0$ であるのに，実際の出力が $\hat{y}(\boldsymbol{x}) = 1$ となった場合である．この場合は出力ユニットへの入力 $u = \sum_i w_i x_i$ の値が大きすぎるということになる．したがって，入力ユニットから信号が小さくなるように，発火している入力ユニット $x_i = 1$ との結合を弱めてやればよいことになる．つまり発火しているユニット x_i との結合定数 w_i をそれぞれ

$$w_i \to w_i - \eta x_i \tag{3.140}$$

と値を更新する．

この 2 つの更新式は，次のように 1 つの式にまとめることができる．

$$w_i \to w_i - \eta(\hat{y}(\boldsymbol{x}, \boldsymbol{w}) - y)x_i \tag{3.141}$$

したがって，訓練サンプルが与えられるたびに，このようにパラメータを更新させていけば，やがてパーセプトロンは目標信号と同じ値を出力するようになると期待できる．この学習手法は 1958 年にローゼンプラットによって提唱され，パーセプトロンの学習則とよばれる．しかし，ローゼンプラットによるパーセプトロンの学習則は線形分離が可能な問題に対しては学習データを十分に与えることでパラメータが最適値に収束していくが，線形分離が不可能な問題の場合は必ずしも収束しないという問題が生じることがわかった．そして，その原因の 1 つは，ユニットの入出力値が 0,1 の 2 値しかとり得ないことにあった．

そこで，パーセプトロンを連続の入出力値をもった現代パーセプトロン (ニューラルネットワーク) に拡張し，その学習を考えることにする．この場合も学習則は (古典) パーセプトロンのときと変わらない．

$$w_i \to w_i - \eta(\hat{y}(\boldsymbol{x}, \boldsymbol{w}) - y)x_i \tag{3.142}$$

ただし，いまの場合は $\hat{y}(\boldsymbol{x}; \boldsymbol{w})$ と y どちらも実数値をとる．訓練サンプルが与えられるたびに，この学習則で重みパラメータを調整していき，正しい出力結果を行えるようにする方法はウィドロウ・ホフ則または，デルタ則とよばれる．デルタ則とよばれるの理由は，訓練信号と出力のズレを表すデルタ $\delta \equiv \hat{y}(\boldsymbol{x} - y)$ という量を用いて重みパラメータを更新しているからである．

これまでは中間層のない現代パーセプトロン (ニューラルネットワーク) の学習を考えてきたが，中間層を追加した多層ニューラルネットワークの場合はどのように学習を行えばよいか．式

(3.142) の形のままでは，最終層以外の重みパラメータをどのように調整すればよいかわからない．したがってすべての層のノードに対してデルタの概念を拡張する必要がある．そのために中間層にある重みパラメータが，どれだけ出力の誤差に寄与しているかを知る必要がある．そこでデルタ則を違う視点から見てみる．実は $(\hat{y}(\boldsymbol{x}) - y)x_i$ という量には，数理的な意味があり，出力層の活性化関数を恒等写像としバイアスを無視すると，出力は $\hat{y}(\boldsymbol{x}) = \sum_i w_i x_i$ となる．これと目標出力の間の二乗誤差は $E = (\hat{y}(\boldsymbol{x} - y)^2/2)$ なので，$(\hat{y}(\boldsymbol{x}) - y)x_i$ はこの誤差のパラメータ w_i による微分にほかならない．つまり，デルタ則は，出力の二乗誤差を極小にするように最適化する勾配降下法にほかならない．したがってパーセプトロンの学習を一般化することで勾配降下法のアイデアに自然にたどり着くのである．そして，多層ニューラルネットワークに勾配降下法を応用させた学習手法が誤差逆伝播法である．

3.9.2 誤差逆伝播法の考え方

勾配降下法を用いた多層ニューラルネットワークの学習手法を誤差逆伝播法とよぶ．ここでは，単純なモデルを用いて誤差逆伝播法の考え方について説明する．ここで扱う勾配とは誤差関数の微分係数

$$\Delta_{\boldsymbol{w}} E(\boldsymbol{w}) = \frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} \tag{3.143}$$

であった．誤差関数 E はモデルの出力と正解値の間のズレを測る量であるため，出力 $\hat{\boldsymbol{y}}(\boldsymbol{x}; \boldsymbol{w})$ を通じてニューラルネットワークのパラメータに依存している．例えば二乗誤差関数の場合は，$E(\boldsymbol{w}) = (\hat{\boldsymbol{y}}(\boldsymbol{x}; \boldsymbol{w}) - \boldsymbol{y})^2/2$ である．つまり，微分のチェインルールを使うと誤差関数の第 l 層にあるパラメータ $w_{ji}^{(l)}$ による微分は

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{ji}^{(l)}} = \sum_{k=1}^{D_l} \frac{\partial E(\boldsymbol{w})}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial w_{ji}^{(l)}} \tag{3.144}$$

と書ける．$\partial E(\boldsymbol{w})/\partial \hat{y}_k$ の値は簡単に計算できるため，この値は $\hat{\boldsymbol{y}}(\boldsymbol{x}; \boldsymbol{w})$ をパラメータ $w_{ji}^{(l)}$ で偏微分する計算に帰着するが，深層ニューラルネットワークにおいてこの計算がかなる複雑になる．

このことを理解するために例として各層が 1 つのユニットからなる L 層の深層ニューラルネットワークという単純なモデルを考える．図 3.19 は第 l 層とその前後にあるユニットに注目したものである．

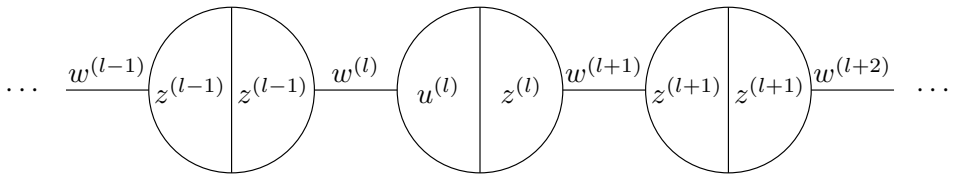


図 3.19: 1 つのユニットからなる層を 1 列につなげたニューラルネットワークの第 l 層に注目した図

この場合，第 l 層の活性は $u^{(l)} = w^{(l)} z^{(l-1)} = w^{(l)} f(u^{(l-1)})$ で出力が $\hat{y} = z^{(L)} = f(u^{(L)})$ であることから，

$$\begin{aligned} y &= f(u^{(L)}) \\ &= w^{(L)} f\left(u^{(L-1)}\right) \\ &= w^{(L)} f\left(w^{(L-1)} f\left(u^{(L-2)}\right)\right) \\ &\quad \vdots \\ &= f\left(w^{(L)} f\left(w^{(L-1)} f\left(\cdots w^{(l+1)} f\left(w^{(l)} f\left(\cdots f(x)\right)\right)\right)\right)\right) \end{aligned} \tag{3.145}$$

となり，これは多数の写像の合成になっているということがわかる．したがって，これを $w^{(l)}$ で微分しようと思う場合，最終の第 L 層から離れれば離れるほど巨大な合成関数を計算して，その微分を実行することになる．このように多層ニューラルネットワークのパラメータ微分は層の深さが原因で計算が複雑になってしまう．計算の複雑さのほかにも，そのまま安直に数値微分として実装するのは精度としても計算量の観点からしても良い方法ではない．ではどうするかというと，我々の手で簡略化できるところは数学的に計算し尽くして，シンプルなアルゴリズムの形に帰着させてから実装するのである．その結果から得られるアルゴリズムが誤差逆伝播法というスマートな手法である

誤差逆伝播法の仕組みを理解するため，まず $u^{(l)} = w^{(l)} z^{(l-1)}$ という関係に注目する．$w^{(l)}$ の変動はまずは $u^{(l)}$ の値に直接影響を与えるため，チェインルールより

$$\frac{\partial E}{\partial w^{(l)}} = \frac{\partial E}{\partial u^{(l)}} \frac{\partial u^{(l)}}{\partial w^{(l)}} = \frac{\partial E}{\partial u^{(l)}} z^{(l-1)} \equiv \delta^{(l)} z^{(l-1)} \tag{3.146}$$

と書き換えられる．したがって $\delta^{(l)} \equiv \partial E / \partial u^{(l)}$ という量がわかれば勾配が決まる．ところが $u^{(l+1)} = w^{(l+1)} f(u^{(l)})$ であるため，このデルタが次の層の $\delta^{(l+1)}$ と関係付く．つまり，

$$\begin{aligned} \delta^{(l)} &= \frac{\partial E}{\partial u^{(l+1)}} \frac{\partial u^{(l+1)}}{\partial u^{(l)}} = \delta^{(l+1)} w^{(l+1)} f'(u^{(l)}) \\ \delta^{(l+1)} &= \frac{\partial E}{\partial u^{(l+2)}} \frac{\partial u^{(l+2)}}{\partial u^{(l+1)}} = \delta^{(l+2)} w^{(l+2)} f'(u^{(l+1)}) \\ &\quad \vdots \\ \delta^{(L-1)} &= \frac{\partial E}{\partial u^{(L)}} \frac{\partial u^{(L)}}{\partial u^{(L-1)}} = \delta^{(L)} w^{(L)} f'(u^{(L)}) \\ \delta^{(L)} &= \frac{\partial E}{\partial u^{(L)}} \end{aligned} \tag{3.147}$$

である．したがって $\delta^{(L)}$ から順に遡って δ の値を計算していけば，誤差関数のパラメータ微分を計算することができる．これが誤差逆伝播法の考えである．

3.9.3 誤差関数の勾配計算

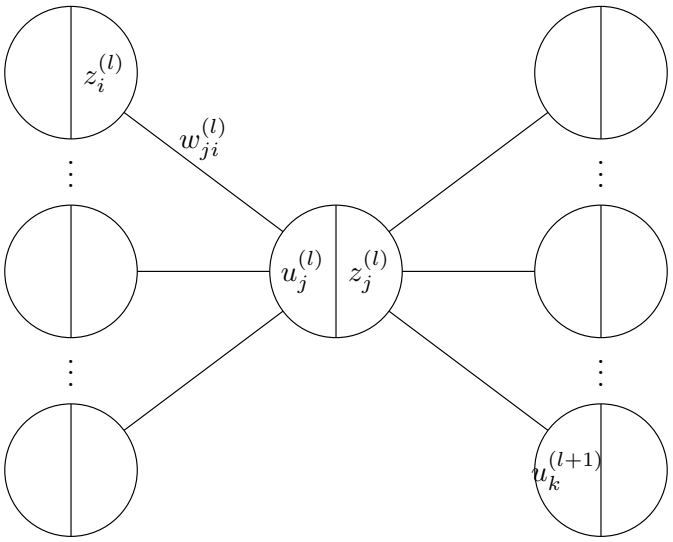


図 3.20: 1 つのユニットからなる層を 1 列につなげたニューラルネットワークの第 l 層に注目した図

誤差逆伝播法に至るためのアイデアの 1 つ目は，巨大な合成関数の微分を一挙に計算せず，困難を分割することであった．そのために注目している重み $w_{ji}^{(l)}$ の変動が，ネットワークの局所的な構造を通じて周囲にどれくらいの影響を与えるか，という観点から勾配を書き換える．図**??**のように，重み $w_{ji}^{(l)}$ の変動は，まず l のユニット j の総入力を変化させる．この総入力 $u_j^{(l)}$ の変動がネットワークを順次伝播していき，出力を変化させ，最終的に誤差関数の変動を与える．この事実を微分操作で表現すると，誤差関数 $E_n(u_j^{(l)}(w_{ji}^{(l)}))$ は総入力 $u_j^{(l)}$ を通じて，重み $w_{ji}^{(l)}$ に間接的に依存しているため，

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$

と書くことができる．2 行目の右辺の 1 つ目の因子をデルタとよび．

$$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial u_j^{(l)}} \tag{3.148}$$

と定義した．これはユニット j からのシグナルが，どれほど最終的に誤差に聞いているかを測る尺度となる量である．したがって，この値が求まれば勾配が決定する．このように勾配をデルタとユニット出力に分けることが第 1 ステップである．

第 2 ステップであるデルタを決めるアルゴリズムは誤差逆伝播法の核となる部分である．実はこの部分も，答えが一度わかってしまえばそれほど難しい話ではない．なぜなら，デルタを満たす漸化式を見つけてやればよいからである．漸化式を導くのみ，デルタの定義に立ち返ってみる．デルタは誤差関数をユニット j への総入力 $u_j^{(l)}$ で微分したものである．そこでこの総入力の変化がど

のように誤差関数に影響をするのかを見るために，再びネットワークの局所的な構造に注目する．図**??**より，ユニット j の総入力 $u_j^{(l)}$ は活性化関数が作用することでこのユニットの出力へ変換され，隣接する $l+1$ 層のユニットたちへ入力する．したがって $u_j^{(l)}$ の変動は，次層のユニットたちの総入力 $u_k^{(l+1)}$ の変化を通じて誤差関数を変動させる．つまり微分係数は

$$\delta_j^{(l)} = \frac{\partial E_n}{\partial u_j^{(l)}} = \sum_{k=0}^{d_{l+1}-1} \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \tag{3.149}$$

という性質を満たす．右辺の 1 つ目の因子は $l+1$ 層でのデルタ $\delta_k^{(l+1)}$ にほかならないため，この式は隣接層間のデルタの間の漸化式である．

$$u_k^{(l+1)} = \sum_{j'} w_{kj'}^{(l+1)} f(u_{j'}^{(l)}) \tag{3.150}$$

であったことを思い出すと，この式は次のようにまとめることができる．

$$\delta_j^{(l)} = \sum_{k=0}^{d_{l+1}-1} \delta_k^{(l+1)} w_{kj}^{(l+1)} f'(u_{j'}^{(l)}) \tag{3.151}$$

これがデルタの逆伝播を記述する式である．逆伝播とよばれる理由は，通常のニューラルネットワークの入力が

$$u_j^{(l)} = \sum_{k=0}^{d_{l+1}-1} w_{kj}^{(l)} f(u_{i'}^{(l-1)}) \tag{3.152}$$

を通じて重み $w_{ji}^{(l)}$ と活性化関数 f を受け取りながら $l-1$ 層から l 層へ順伝播するのに対し，式 3.152 を見ると，デルタという量が $w_{kj}^{(l+1)}$ 倍の作用を受けて $l+1$ 層から l 層へ逆向きに伝播しているからである．実際にデルタを計算するアルゴリズムにおいても，出力層で計算されたデルタの「初期値」の情報を入力層側に向けて順次逆向きに伝播させている．したがって，実際の計算では，まず入力を出力層まで順伝播させ，出力層での誤差を計算させる．そして，その結果をもとにデルタを逆伝播させる．

$$\begin{aligned} &\boldsymbol{x}_n = \boldsymbol{z}^{(1)} \to \boldsymbol{z}^{(2)} \to \cdots \to \boldsymbol{z}^{(L-1)} \to \boldsymbol{z}^{(L)} = \hat{\boldsymbol{y}} \\ &\boldsymbol{\delta}^{(1)} \leftarrow \boldsymbol{\delta}^{(2)} \leftarrow \cdots \leftarrow \boldsymbol{\delta}^{(L-2)} \leftarrow \boldsymbol{\delta}^{(L-1)} \leftarrow \boldsymbol{\delta}^{(L)} \end{aligned} \tag{3.153}$$

$\boldsymbol{\delta}^{(l)}$ は l 層のデルタを並べて作ったベクトルである．来れたの結果を式 3.148 の形に組み合わせることですべての勾配が決まる．

$$\frac{\partial E_n}{\partial \boldsymbol{w}_{ji}^{(l)}} = \boldsymbol{\delta}^{(l)} \left(\boldsymbol{z}^{(l-1)}\right)^\top \tag{3.154}$$

これが誤差逆伝播法である．

3.9.4 逆伝播計算の初期値

逆伝播において漸化式を解くための初期値にあたるのは，出力層におけるデルタである．この値は簡単に計算できる．というのも出力層の活性化関数を f とすると，最終出力は $\hat{y}_j = f(u_j^{(L)})$ であるため，出力層のユニット j のデルタは $E_n(\hat{y}_j(u_j^{(L)}))$ を $u_j^{(L)}$ で微分して，

$$\delta_j^{(L)} = \frac{\partial E_n}{\partial \hat{y}_j} f'(u_j^{(L)}) \tag{3.155}$$

となるからである．例えば回帰問題などの場合には誤差関数として二乗誤差 $E_n = (\hat{\boldsymbol{y}}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y})^2/2$ が用いられる．この場合デルタの具体形は

$$\delta_j^{(L)} = (\boldsymbol{y}_j - y_j)\hat{y}'_j(u_j^{(L)}) \tag{3.156}$$

である．出力層の活性化関数が恒等写像であれば，当然ながら $\delta_j^{(L)} = (\hat{y}_j - y_j)$ であり，前節のデルタ則のパラメータ更新 (**??**) を再現する．

一方，誤差関数が交差エントロピー (**??**) である場合，

$$\delta_j^{(L)} = -\sum_k \frac{t(y_n)_k}{\hat{y}_k} \hat{y}'_k(u_j^{(L)}) \tag{3.157}$$

であった．この出力層の活性化関数はソフトマックス $y_k = e^{u_k^{(L)}} / \sum_{k'} e^{u_{k'}^{(L)}}$ である．ソフトマックスの微分はクロネッカーのデルタを用いると

$$\frac{\partial \hat{y}_k}{\partial u_j^{(L)}} = \delta_{kj}\hat{y}_k - \hat{y}_j\hat{y}_k \tag{3.158}$$

と書き換えられるという性質がある．さらにクラス分類のときは，どのような訓練サンプルも $\sum_k t_k = 1$ という性質を満たしているため，デルタは結局 $\delta_j^{(L)} = (\hat{y}_j - t_j)$ という先ほどと同じ形に帰着する．

3.9.5 勾配計算

勾配計算の目的はニューラルネットワークの勾配法による学習であったので，誤差逆伝播で終わりではない．各訓練サンプルに対して計算された勾配 $\partial E_n/\partial w_{ji}^{(l)}$ を，学習に用いる (ミニ) バッチ $\mathcal{D}$ で平均して実際のパラメータの更新量を求める必要がある．

$$\Delta w_{ji}^{(l)} = -\eta \frac{\partial E}{\partial w_{ji}^{(l)}} = -\eta \frac{1}{|\mathcal{D}|} \sum_{n\in\mathcal{D}} \frac{\partial E}{\partial w_{ji}^{(l)}} \tag{3.159}$$

そのうえで，各時刻 t でのパラメータ値で計算した $\Delta w_{ji}^{(t,l)}$ を使って

$$w_{ji}^{(t+1,l)} = w_{ji}^{(t,l)} + \Delta w_{ji}^{(t,l)} \tag{3.160}$$

と重みパラメータを更新する．その後再び順伝播と逆伝播を行い，次時刻にも同じ操作を繰り返す．これをパラメータの収束まで繰り返すのが勾配降下法による学習であった．

3.9.6 デルタの意味

ここでは誤差逆伝播法に現れたデルタの意味について説明する．出力層におけるデルタには $\delta_j^{(L)} = \hat{y}_j(\boldsymbol{x}; \boldsymbol{w}) - y_j$ という形をしていた．これは訓練サンプル $\boldsymbol{x}$ を入力した際の最終出力 $\boldsymbol{y}$ と，訓練サンプルの目標信号 $\boldsymbol{y}$ の差なので，まさにニューラルネットワークによる推論の誤差である．勾配降下法はこの誤差をできるだけ打ち消すように，出力層とその手前の層の間の重みパラメータ $w_{pq}^{(L)}$ を修正する．つまり，このデルタ $\boldsymbol{\delta}^{(L)}$ は，出力層ノードにつながる重みがどの程度誤差に寄与しているかを測る量になっている．では，中間層のノードたちがどのくらい誤差関数に効いているかを測っているのでしょうか．

中間層のノードの総入力を

$$u_j^{(l)} \to u_j^{(l)} + \Delta u_j^{(l)} \tag{3.161}$$

と変動させてみる．デルタの定義から，この変動により誤差関数はだいたい

$$\Delta E \approx \delta_j^{(l)} \Delta u_j^{(l)} \tag{3.162}$$

だけズレる．もしこのデルタ係数がほとんど 0 に近ければ，このノードの総入力の大きさを調整してもほとんど誤差関数の大きさは改善できないことになる．つまりこのノードへの総入力を決めている重みたちは，ほとんど誤差に寄与しないということになる．

一方デルタの絶対値が大きい場合は，このノードの総入力を変えることで誤差関数は劇的に変動する．つまり，パラメータ空間の中で，現在地点から少しずれるだけで大きく誤差関数が変化しえるため，現在のニューラルネットワークは最適値よりかなり上にいる可能性があるといえる．つまりこのノードの総入力は，誤差関数を極小化しない不適切な値を示しているということであるため，誤差を減らすようにこのノードへの入力重み $w_{ji}^{(l)}$ を修正すべきである．この総入力の変動と逆符号の方向に変動を与えるパラメータの修正をする．これが誤差逆伝播法をもとにした勾配降下法を行っていることにほかならない．このように一般化デルタ $\delta_j^(l)$ は，l 層のノード j に付随した局所的な誤差を測る量になっている．

第 4 章

機械学習によるイジング模型の相転移検出

4.1 イジング模型のモンテカルロ法

モンテカルロ法とは乱数を使って積分や和を行う手法である．ここでは，期待値の和を乱数を用いて調べる．表記の都合上，今後スピン配位を $C(=\{\boldsymbol{\sigma}\})$ と表記することにする．あるスピン配位 C の実現確率を

$$P(C) = \frac{1}{Z} e^{-\beta E[C]} \tag{4.1}$$

として確率的にスピン配位 C を生成して，生成したスピン配位 C を使った平均を用いて期待値を推定する手法をとる．

しかし，スピン配位 C の実現確率 (4.1) は直接計算できない．なぜなら，分母の分配関数 Z を計算するために，結局全ての配位の和を計算する必要があるからである．そこで，分配関数 Z の計算を避けてスピン配位を生成できる手法であるマルコフ連鎖モンテカルロ法 (Markov chain Monte-Carlo) を以下で説明する．

4.1.1 マルコフ連鎖モンテカルロ法

今，調べたい期待値は一般的に

$$\mathbb{E}[f(C)] = \sum_C f(C)P(C) \tag{4.2}$$

と書ける．ここで $f(C)$ は配位の関数で，例えば磁化率などである．また，$\sum_C$ は可能なスピン配位についての足し上げを表す．格子点の数が K 個であるとき，イジングモデルなら 2^K 個の足し算になる．これを

$$\mathcal{C} = \{C_1, C_2, C_3, \cdots\} \tag{4.3}$$

という 2^K よりもずっと短い列の平均に置き換え，

$$\mathbb{E}[f(C)] \approx \frac{1}{|\mathcal{C}|}\sum_{C_k\in\mathcal{C}} f(C_k) \tag{4.4}$$

を代わりに評価したい．ここで $|\mathcal{C}|$ は配位の数を表す．当然，適当に作った列では正しい期待値を与えないが，以下を満たす $P(C_a|C_b)$ に従って C を生成すれば正しい期待値を与えることが知られている．

$$P(C_a|C_b)P(C_b) = P(C_b|C_a)P(C_a) \tag{4.5}$$

この証明は大数の法則による．ここで条件付き確率 $P(C_a|C_b)$ は遷移確率とよばれ，この式は詳細釣り合い (detailed balance) の式と呼ばれている．乱数を使ったアルゴリズムをモンテカルロ法と呼ぶが，上のような確率的に配位の列を作るアルゴリズムをマルコフ連鎖モンテカルロ法という．

4.1.2 イジングモデルに対するメトロポリス法

マルコフ連鎖モンテカルロ法にもいくつかの手法が存在し，その中でとくによく用いられているものとして，メトロポリス法と熱浴法があげられる．ここではメトロポリス法について説明する．

ある配位 C_b と C_a を比較したときに C_a のエネルギーの方が低い場合，つまり $E(C_b) > E(C_a)$ を考える．この場合は C_a の方がエネルギー的に安定なので，C_b から C_a への遷移確率を

$$P(C_a|C_b) = 1 \tag{4.6}$$

ととることにする．このとき，詳細釣り合いの式は

$$P(C_b) = P(C_b|C_a)P(C_a) \tag{4.7}$$

となる．これより，C_a から C_b の遷移確率が決まってしまい，

$$P(C_b|C_a) = \frac{P(C_b)}{P(C_a)} = \frac{e^{-\beta E(C_b)}/Z}{e^{-\beta E(C_a)}/Z} = e^{-\beta\Big(E(C_b)-E(C_a)\Big)} \tag{4.8}$$

となる．したがって，C_a から C_b の遷移確率として，

$$P(C_b|C_a) = \begin{cases} \exp\Big[-\beta\Big(E(C_b)-E(C_a)\Big)\Big], & E(C_b) > E(C_a) \\ 1, & E(C_b) < E(C_a) \end{cases} \tag{4.9}$$

ととると詳細釣り合いの式を満たすことになる．このアルゴリズムをメトロポリス法と呼ぶ.

メトロポリス法をイジングモデルに適用する場合の手順をまとめる．

1. 適当な配位 C_1 を用意する．
2. 以下を $i = 1, 2, 3, \cdots, N_{\rm conf}$ まで繰り返す，
   (a) C_i のあるサイトのスピンを反転し，その配位を C_i' とおく．
   (b) エネルギー $E(C_i)$ と $E(C_i')$ を計算する．

（c）$E(C_i) > E(C'_i)$ なら $C_{i+1} = C'_i$ とし，$i \to i+1$ とおいて (a) へ戻る．(受理 (accept))
（d）乱数 $r \in [0,1]$ を用意し，$\Delta E = E(C'_i) - E(C_i)$ に対して，$r < \exp(-\beta \Delta E)$ ならば $C_{i+1} = C'_i$ とし，$i \to i+1$ とおいて (a) へ戻る．(受理 (accept))
（e）$C_{i+1} = C_i$ とし，$i \to i+1$ とおいて (a) へ戻る．(棄却 (reject))

ここで,「棄却」は提案された配位を使わずに前の配位をそのまま次の配位として受け入れるという意味であることに注意してほしい.

上記の手順に従うと，配位の列

$$\mathcal{C} = \{C_1, C_2, C_3, \cdots, C_{N_{\rm conf}}\} \tag{4.10}$$

が得られる．しかし最初の方の配位は適当に選んだ初期配位の影響を受けているため，期待値の計算に含めてはいけない．例えば $N_{\rm disc}(1 < N_{\rm disc} << N_{\rm conf})$ 番目 (disc は discard の略) まで除いたとすると,

$$\mathbb{E}[f(C)] \approx \frac{1}{N_{\rm disc} - N_{\rm conf}} \sum_{k=N_{\rm disc}}^{N_{\rm conf}} f(C_k) \tag{4.11}$$

のように期待値を計算できる．$N_{\rm disc}$ は配位ごとの物理量を観察して初期配位の影響がなくなったあたりの番号をとればよい.

4.1.3 イジングモデルの熱浴法

次に熱浴法 (heatbath algorothm) をみる．このアルゴリズムは機械学習の文脈ではギブスサンプリング (Gibbs sampling) と呼ばれる．熱浴法は，ある 1 自由度を着目し，その周りの自由度を熱浴とみなして更新するマルコフ連鎖モンテカルロ法の一種である.

熱浴法は条件付き確率から導出される．イジングモデルのあるサイト j に着目し，そのサイト以外を熱浴をみなしたときにサイト j がどうなるかを条件付き確率に従って決めるのである.

例えば,

$$C_j\text{: サイト } j \text{ にあるスピンの状態} \tag{4.12}$$
$$\overline{C}_j\text{: サイト } j \text{ 以外のスピンの状態} \tag{4.13}$$

とすると，$\overline{C}_j$ の下での C_j の条件付き確率は

$$P(C_j|\overline{C}_j) = \frac{P(C_j, \overline{C}_j)}{P(\overline{C}_j)} \tag{4.14}$$

と書ける．ある配位を与える確率を

$$\begin{aligned} P(C) &= P(C_j, \overline{C}_j) \\ &= \frac{1}{Z} \exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k + (\sigma_j \text{を含まない項}) \right] \end{aligned} \tag{4.15}$$

と書く．ここで，サイト j の最近接サイトをまとめて $\mathcal{N}(j)$ と書いた．

C_j の状態 (サイト j のスピン状態) を足し上げて周辺化した確率 $P(\overline{C}_j)$ は，$P(\overline{C}_j) = \sum_{C_j} P(C_j, \overline{C}_j)$ を使うと A_j は $\sigma_j = \{+1, -1\}$ をとるので，全て足し上げて

$$P(\overline{C}_j) = \frac{1}{Z} \sum_{\sigma_j = \pm 1} \exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k + (\sigma_j \text{を含まない項}) \right] \tag{4.16}$$

とわかる．ここから条件付き確率 $P(C_j|\overline{C}_j)$ は，定義 $P(C_j|\overline{C}_j) = P(C_j, \overline{C}_j)/P(\overline{C}_j)$ から

$$\begin{aligned} P(C_j|\overline{C}_j) &= \frac{1}{Z} \exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k + (\sigma_j \text{を含まない項}) \right] \Bigg/ \\ &\qquad \frac{1}{Z} \sum_{\sigma_j = \pm 1} \exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k + (\sigma_j \text{を含まない項}) \right] \\ &= \frac{\exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k \right]}{\sum_{\sigma_j = \pm 1} \exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k \right]} \\ &= \frac{\exp\left[-\beta J \sigma_j \sum_{k \in \mathcal{N}(j)} \sigma_k \right]}{\exp\left[-\beta J \sum_{k \in \mathcal{N}(j)} \sigma_k \right] + \exp\left[+\beta J \sum_{k \in \mathcal{N}(j)} \sigma_k \right]} \end{aligned} \tag{4.17}$$

となる．最後の式を見ると，サイト j の周りのスピン配位だけでサイト j の状態が決まるということがわかる．また，ここでもメトロポリス法のときと同様に分配関数 Z の計算が必要なくなっている．

例えば，サイト j のスピンが $+1$ である場合は

$$P(s_j = +1) = \frac{\exp\left[\beta J \sum_{k \in \mathcal{N}(j)} s_k \right]}{\exp\left[\beta J \sum_{k \in \mathcal{N}(j)} s_k \right] + \exp\left[-\beta J \sum_{k \in \mathcal{N}(j)} s_k \right]} \tag{4.18}$$

となる．なのでこの確率に従って次の配位におけるサイト j のスピンを決定 (更新) すればよい．この確率の評価はメトロポリス法の乱数を使っている個所と同じように乱数を用いて評価すればよい．また $P(s_j = -1|B) = 1 - P(s_j = +1|B)$ である．そして空間全部のスピンについて繰り返せば次の配位が得られる．

4.1.4 温度ラベルを持つ配位データの生成

4.2 相の分類器による相転移検出

カラスクイラ (Juan Carrasquilla) とメルコ―(Roger G.Melko) は，2016 年の 5 月に”Machine learning phases of matter”を投稿し，後に Nature Physics 誌に掲載された．彼らの研究によってはじめてニューラルネットワークを用いて相転移が検出可能であることが示された．ここでは，

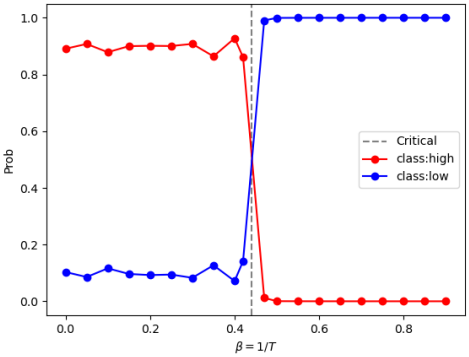
この論文の内容を引用して，2 次元正方イジング模型と 2 次元三角イジング模型の相転移の検出を行う．

4.2.1 学習データの作成

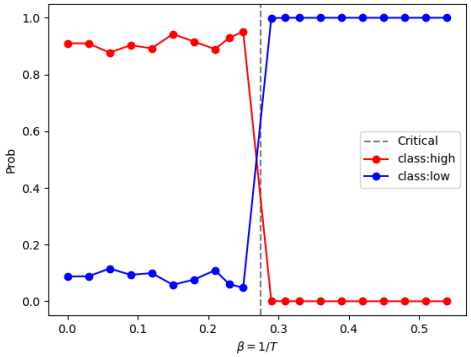
ニューラルネットワークを用いて学習を行うにはまず，モデルの学習のためのデータセットを作成する必要がある．相の分類器の作成には，スピン配位とその配位が秩序相か無秩序相かを示す教師ラベルをセットにした教師ありデータを大量に用意する．

この教師データの作成は python を用いて次の手順で生成した．

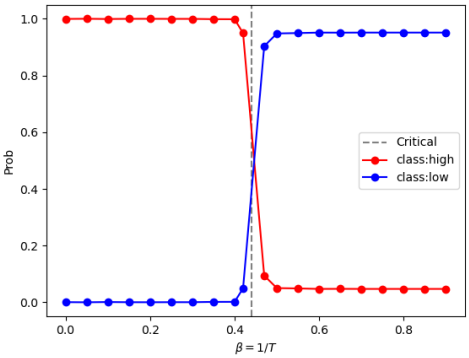
1. 温度 T を選ぶ
2. メトロポリス法を用いて温度 T のスピン配位 $\boldsymbol{\sigma}$ をサンプルする
3. $T < 2.27$ の場合 $d = 0$，$T < 2.27$ の場合 $d = 0$ とする
4. $(\boldsymbol{\sigma}, d)$



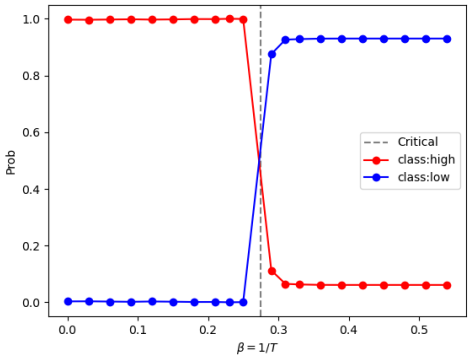
(a) 正方イジング模型



(b) 三角イジング模型



(a) 正方イジング模型



(b) 三角イジング模型

4.3 温度測定器による相転移検出

ニューラルネットワークで相の分類器を作ることで，モデルに秩序相と無秩序相の違いを学習させることがわかった．しかし，この相の分類器作成では，モデルに相転移の情報を与える必要があった．つまり，事前に系の相転移温度がわかっている場合でしか．この検出器は作れないということになる．これでは，使える場面が限られてしまうので，どうにかしてモデルに相転移温度の情報を与えないで相の検出を行えるようにしたい．

そこで，相の分類器ではなく，温度測定器を作るという視点でモデルを学習させてみる．

4.4 相転移検出がなぜ可能なのか？

まとめ

研究のまとめを書く.

謝辞

この修士論文を完成させるにあたり、多くの方々に支えられ、助言をいただきました。まず初めに、素粒子論研究の阪口教授，百武教授，藤原教授，そして山下助教に心より感謝申し上げます。機械学習ゼミにおいて、多大なるお世話になりました。お陰様で、機械学習やディープラーニングの基礎知識を深めることができ、研究において大きな成果を得ることができました。

特に、担当教員である藤原教授には、修論ゼミにおいて大変お世話になりました。緻密なアドバイスや示唆により、論文の進行や内容の向上に大いに貢献いただきました。深い感謝の意を表します。

また，同じ研究室の伊藤君には、研究分野が近しいこともあり、数々の相談に乗っていただきました。伊藤君の専門知識と経験から得られたアドバイスは、研究の方向性を明確にする上で非常に有益でした。心から感謝いたします。

最後に、私の研究活動において支えてくださったすべての関係者に感謝いたします。皆様のご協力なしでは、この論文の完成はあり得ませんでした。改めて、心より感謝いたします。

参考文献

[1] Tom Michael Mitchell. Machine Learning. Mc Graw-Hill, 1997.

付録 A

aaa

aaa

A.1　bbb

bbb