

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 藤原高德

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	3
1.1	研究背景	3
1.2	研究目的	3
第 2 章	イジング模型の相転移	4
2.1	相転移, 臨界現象とは	4
2.2	強磁性イジング模型	5
2.3	一次元イジング模型	9
2.4	二次元イジング模型	12
2.5	二次元イジング模型の厳密解の計算	12
第 3 章	機械学習と深層学習	19
3.1	機械学習とは	19
3.2	深層学習とは	21
3.3	統計入門	21
3.4	機械学習の基礎	27
3.5	ニューラルネットワーク	34
3.6	ニューラルネットワークによる機械学習	41
3.7	畳み込みニューラルネットワーク	44
3.8	勾配降下法	44
3.9	改良された勾配降下法	47
3.10	誤差逆伝播法	51
第 4 章	機械学習によるイジング模型の相転移検出	52
4.1	配位データの生成方法	52
4.2	相の分類器による相転移検出	52
4.3	温度測定器による相転移検出	52
4.4	相転移検出がなぜ可能なのか?	52
	謝辞	54

参考文献	55
付録 A aaa	56
A.1 bbb	56

第 1 章

はじめに

1.1 研究背景

1.2 研究目的

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°C 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°C 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，水と氷が共存するのはちょうど 0°C に限られ，水と水蒸気が共存するのはちょうど 100°C 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°C から 100°C 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い．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、教師全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トツである。

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 $i = 1, 2, \dots, N$ 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ののっている。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 $\{\sigma\} = (\sigma_1, \sigma_2, \dots, \sigma_N)$ 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。それぞれのスピンの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 $\{\sigma\}$ に対応するエネルギー固有値を

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

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、

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

となる。つまり、隣り合う格子点のスピンの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

$$Z_L(\beta, H) = \sum_{(\sigma_1, \dots, \sigma_N)} \exp[-\beta E_{(\sigma_1, \dots, \sigma_N)}] \quad (2.3)$$

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

$$\langle \hat{g} \rangle_{\beta, H}^{\text{can}} := \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1, \dots, \sigma_N)} g_{(\sigma_1, \dots, \sigma_N)} \exp[-\beta E_{(\sigma_1, \dots, \sigma_N)}] \quad (2.4)$$

である。ここで、 $g_{(\sigma_1, \dots, \sigma_N)}$ は状態 $(\sigma_1, \dots, \sigma_N)$ 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

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

，物理量としての磁化を

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

と定義する。このとき、磁化の期待値は

$$\begin{aligned} m_L(\beta, H) &:= \langle \hat{m} \rangle_{\beta, H}^{\text{can}} \\ &= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1, \dots, \sigma_N)} m_{(\sigma_1, \dots, \sigma_N)} \exp[-\beta E_{(\sigma_1, \dots, \sigma_N)}] \\ &= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1, \dots, \sigma_N)} \frac{1}{N} \sum_{j=1}^N \mu_0 \sigma_j \exp[-\beta E_{(\sigma_1, \dots, \sigma_N)}] \\ &= \frac{1}{Z_L(\beta, H)} \sum_{(\sigma_1, \dots, \sigma_N)} \frac{1}{\beta N} \frac{\partial}{\partial H} \exp[-\beta E_{(\sigma_1, \dots, \sigma_N)}] \\ &= \frac{1}{\beta N} \frac{1}{Z_L(\beta, H)} \frac{\partial}{\partial H} Z_L(\beta, H) \\ &= \frac{1}{\beta N} \frac{\partial}{\partial H} \ln Z_L(\beta, H) \\ &= -\frac{\partial}{\partial H} f_L(\beta, H) \end{aligned} \quad (2.7)$$

と書ける。これ以降、期待値 $\langle \hat{m} \rangle_{\beta, H}^{\text{can}}$ 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

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

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、一つのスピンの着目すれば、 $\sigma_i = \sigma_j$ 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、すべてのスピンの値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。すべてのスピンの値が $+1$ の状態での磁化は μ_0 、すべてのスピンの値が -1 の状態での磁化は $-\mu_0$ なので、

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

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 $(\sigma_1, \dots, \sigma_N)$ 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 $(\tilde{\sigma}_1, \dots, \tilde{\sigma}_N)$ 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

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

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 $(\sigma_1, \dots, \sigma_N)$ について足し上げることは, すべての $(\tilde{\sigma}_1, \dots, \tilde{\sigma}_N)$ 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,

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

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

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

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

$$Z = \sum_{(\sigma_1, \dots, \sigma_N)} \exp \left[\beta J \sum_{i=1}^N \sigma_i \sigma_{i+1} + \beta h \sum_{i=1}^N \sigma_i \right] \tag{2.15}$$

となる. ここで, 次のような対称行列 T を定義する.

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

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

このようにすることで、分配関数 (2.15) を

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

を書くことができる。行列の積の定義より、

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

であることを用いると、分配関数は

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

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 は適当な直行列 O を用いて、

$$O^{-1} T O = \begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix} \quad (2.21)$$

と対角化できる。転送行列 T の固有値は

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

より、(2.20) はさらに

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

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、

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

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 $\beta \rightarrow \infty$ 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の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．

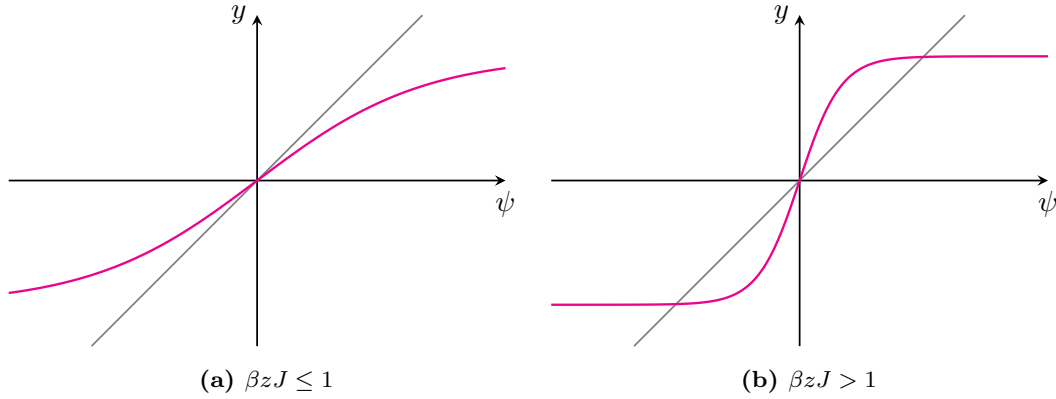


図 2.3: $h = 0$ での自己整合方程式 $\psi = \tanh(\beta zJ\psi)$

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

$$Z = \text{Tr} \prod_{\langle i,j \rangle} \exp(\beta J \sigma_i \sigma_j) \quad (2.28)$$

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

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

を用いれば,

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

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。

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

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

$$v \sigma_i \sigma_j \longleftrightarrow \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} \quad (2.32)$$

$$v^2 \sigma_i \sigma_j \sigma_k \sigma_l \longleftrightarrow \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} \quad \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} k \\ l \end{array}, \quad v^2 \sigma_i \sigma_j^2 \sigma_k \longleftrightarrow \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} \quad \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} k \end{array} \quad (2.33)$$

$$Z = (\cosh \beta J)^{N_B} \text{Tr} \left[1 + \sum \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} + \sum \left(\begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} + \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} k \\ l \end{array} + \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} i \\ j \end{array} \quad \begin{array}{c} \bullet \\ | \\ \bullet \end{array} \begin{array}{c} k \end{array} \right) + \cdots \right] \quad (2.34)$$

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 \$\alpha\$ 列にあるスピンをまとめて次のように表す。

$$\mu_\alpha \equiv \{\sigma_1, \sigma_2, \dots, \sigma_n\} \quad (2.35)$$

ここで、\$\mu\$ について次の境界条件が成り立つ

$$\mu_{n+1} = \mu_1 \quad (2.36)$$

また、スピン配位全体を \$\{\mu_1, \dots, \mu_n\}\$ で表す。イジング模型では、\$\alpha\$ 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、\$\alpha\$ 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。

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

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

ここで、\$\mu\$ と \$\mu'\$ はそれぞれ、隣り合った列のスピン座標の集まりである。

$$\begin{aligned} \mu &\equiv \{s_1, \dots, s_n\} \\ \mu' &\equiv \{s'_1, \dots, s'_n\} \end{aligned} \quad (2.39)$$

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

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

となり、分配関数は

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

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。

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

これを用いると分配関数は

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

$$= \sum_{\mu_1} \langle \mu_1 | P^n | \mu_1 \rangle = \text{Tr } P^n \quad (2.44)$$

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．

$$P = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{2^n} \end{bmatrix} \quad (2.45)$$

ここで， $\lambda_1, \lambda_2, \dots, \lambda_{2^n}$ 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 $(\lambda_1)^n, (\lambda_2)^n, \dots, (\lambda_{2^n})^n$ である．したがって，分配関数は

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

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． λ_{\max} を P の最大固有値とすると，

$$\lim_{n \rightarrow \infty} \frac{1}{n} \ln \lambda_{\max} = \text{finite number} \quad (2.47)$$

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，

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

もしくは

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

となる．したがって，はさみうちの原理より，

$$\lim_{N \rightarrow \infty} \frac{1}{N} \ln Z = \lim_{n \rightarrow \infty} \frac{1}{n} \ln \lambda_{\max} \quad (2.50)$$

が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 $n \rightarrow \infty$ 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

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

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.

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

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

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

$$\tag{2.55}$$

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.

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

行列乗算の通常の意味で, つまり

$$\langle s_1, \dots, s_n | P | s'_1, \dots, s'_n \rangle \tag{2.57}$$

$$= \sum_{s''_1, \dots, s''_n} \sum_{s'''_1, \dots, s'''_n} \langle s_1, \dots, s_n | V_3 | s''_1, \dots, s''_n \rangle \tag{2.58}$$

$$\times \langle s''_1, \dots, s''_n | V_2 | s'''_1, \dots, s'''_n \rangle \langle s'''_1, \dots, s'''_n | V'_1 | s'_1, \dots, s'_n \rangle \tag{2.59}$$

$$= \sum_{s''_1, \dots, s''_n} \sum_{s'''_1, \dots, s'''_n} \delta_{s_1 s''_1} \dots \delta_{s_n s''_n} \delta_{s''_1 s'''_1} \dots \delta_{s''_n s'''_n} \prod_{k=1}^n e^{\beta H s_k} e^{\beta \epsilon s''_k s'''_{k+1}} e^{\beta \epsilon s'''_k s'_k} \tag{2.60}$$

$$= \sum_{s''_1, \dots, s''_n} \delta_{s_1 s''_1} \dots \delta_{s_n s''_n} \prod_{k=1}^n e^{\beta H s_k} e^{\beta \epsilon s''_k s'''_{k+1}} e^{\beta \epsilon s''_k s'_k} \tag{2.61}$$

$$= \prod_{k=1}^n e^{\beta H s_k} e^{\beta \epsilon s_k s_{k+1}} e^{\beta \epsilon s_k s'_k} \tag{2.62}$$

2.5.4 行列の直積

ここで, 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 $1, 2, \dots, m$ 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

$$\langle ii'|A \times B|jj'\rangle \equiv \langle i|A|j\rangle \langle i'|B|j'\rangle \quad (2.63)$$

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

$$\langle ii' \dots i''|A \times B \times \dots \times C|jj' \dots j''\rangle \equiv \langle i|A|j\rangle \langle i'|B|j'\rangle \dots \langle i''|C|j''\rangle \quad (2.64)$$

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．

$$(A \times B)(C \times D) = (AC) \times (BD) \quad (2.65)$$

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．

$$(A \times B \times \dots \times C)(D \times E \times \dots \times F) = (AD) \times (BE) \times \dots \times (CF) \quad (2.66)$$

2.5.5 スピン行列

ここで, V_1, V_2, 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．

$$X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.67)$$

これらパウリ行列は次の関係式を満たす．

$$\begin{aligned} X^2 &= Y^2 = Z^2 = 1 \\ XY + YX &= 0, \quad YZ + ZY = 0, \quad ZX + XZ = 0 \\ XY &= iZ, \quad YZ = iX, \quad ZX = iY \end{aligned} \quad (2.68)$$

ここで, $2^n \times 2^n$ 行列 $X_\alpha, Y_\alpha, Z_\alpha$ を定義する．

$$\begin{aligned} X_\alpha &\equiv 1 \otimes 1 \otimes \dots \otimes X \otimes \dots \otimes 1 \\ Y_\alpha &\equiv 1 \otimes 1 \otimes \dots \otimes Y \otimes \dots \otimes 1 \\ Z_\alpha &\equiv 1 \otimes 1 \otimes \dots \otimes Z \otimes \dots \otimes 1 \end{aligned} \quad (2.69)$$

$\alpha \neq \beta$ のとき以下の関係式が成り立つ．

$$\begin{aligned} [X_\alpha, X_\beta] &= [Y_\alpha, Y_\beta] = [Z_\alpha, Z_\beta] = 0 \\ [X_\alpha, Y_\beta] &= [X_\alpha, Z_\beta] = [Y_\alpha, Z_\beta] = 0 \end{aligned} \quad (2.70)$$

$2^n \times 2^n$ 行列 $X_\alpha, Y_\alpha, Z_\alpha$ は (2.68) のすべての関係式を満たす．

2.5.6 行列 V'_1, V_2, 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.

$$V'_1 = \mathbf{a} \times \mathbf{a} \times \cdots \times \mathbf{a} \quad (2.71)$$

ここで,

$$\langle s | \mathbf{a} | s' \rangle = e^{\beta \epsilon s s'} \quad (2.72)$$

である. したがって,

$$\mathbf{a} = \begin{bmatrix} e^{\beta \epsilon} & e^{-\beta \epsilon} \\ e^{-\beta \epsilon} & e^{\beta \epsilon} \end{bmatrix} = e^{\beta \epsilon} + e^{-\beta \epsilon} X \quad (2.73)$$

である. 式 (??) を用いることで以下を得る

$$\mathbf{a} = \sqrt{2 \sinh(2\beta \epsilon)} e^{\theta X} \quad (2.74)$$

ここで,

$$\tanh \theta \equiv e^{-2\beta \epsilon} \quad (2.75)$$

である. したがって

$$V'_1 = [2 \sinh(2\beta \epsilon)]^{\frac{1}{2}} e^{\theta X} \times e^{\theta X} \times \cdots \times e^{\theta X} \quad (2.76)$$

$$e^{\theta X} \times e^{\theta X} \times \cdots \times e^{\theta X} = e^{\theta X_1} e^{\theta X_2} \cdots e^{\theta X_n} = e^{\theta(X_1 + X_2 + \cdots + X_n)} \quad (2.77)$$

$$V'_1 = [2 \sinh(2\beta \epsilon)]^{\frac{1}{2}} V_1 \quad (2.78)$$

$$V_1 = \prod_{\alpha=1}^n e^{0x_\alpha}, \quad \tanh \theta \equiv e^{-2\beta \epsilon} \quad (2.79)$$

$$V_2 = \prod_{\alpha=1}^n e^{\beta \epsilon \mathbf{Z}_\alpha \mathbf{Z}_{\alpha+1}} \quad (2.80)$$

$$V_3 = \prod_{\alpha=1}^n e^{\beta B \mathbf{Z}_\alpha}, \quad \mathbf{Z}_{n+1} \equiv \mathbf{Z}_1 \quad (2.81)$$

$$P = [2 \sinh(2\beta \epsilon)]^{\frac{1}{2}} V_3 V_2 V_1 \quad (2.82)$$

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 機械学習とは

機械学習 (machine learning) とは，人間がこなすような学習や知的作業を計算機に実行させるためのアプローチの研究，あるいはその手法そのもののことを意味する．機械学習では，知識を人間が直接アルゴリズムに具体的に書き込んだり教え込んだりせず，データという具体例の集まりから計算機に自動的に学ばせるという方法をとる．

機械学習の定義として T. M. ミッチェル (Tom Michael Mitchell) の書籍 [1] で書かれている定義が有名である．それは

”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

である．ここで，タスク T とは解きたい問題のこと，パフォーマンス評価尺度 P は精度，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．

機械学習では，何かしらのモデルを構築する必要がある．このモデルというのは，なにかしらの入力を与えられたときに出力を返すものであり，その正体は多数のパラメータを持った非線形関数 $f_{\theta}(x)$ ， $\theta = \{\theta_1, \theta_2, \theta_3, \dots\}$ である．このパラメータの値は最初はランダムに初期化されており，

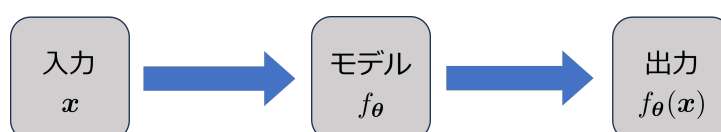


図 3.1: モデルの概略図

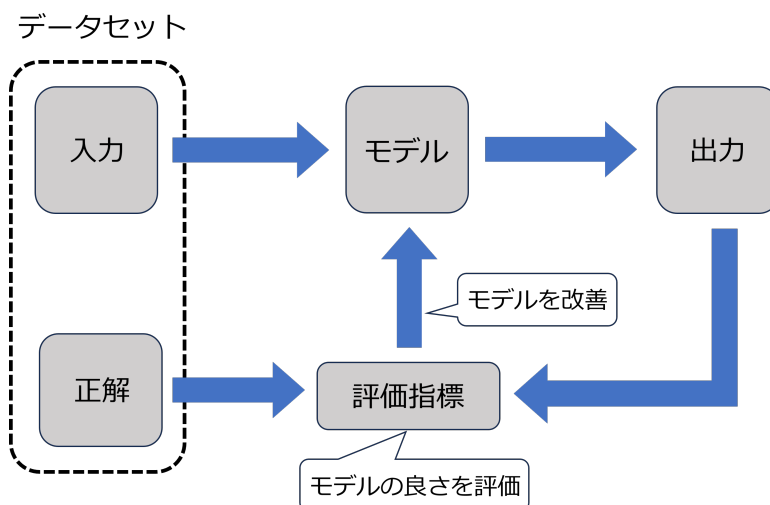


図 3.2: 機械学習の大まかな流れ

入力を与えると出力が返ってくるが、このときの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していく。この一連の流れを繰り返し行うことで、モデルを改善して、正解に近い出力結果を得られるモデルを作成すること機械学習とよぶ。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。

メトロポリス法とメトロポリス・ヘイスティング法は同じですか

3.1.1 代表的なタスク

ここでは、機械学習で解きたい問題であるタスクの代表的な例として「クラス分類」と「回帰」を紹介する。

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。例えば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は、2 クラス分類と呼べる。ここでのクラスというのは「スパムメール or 通常メール」といった分類先のことを指す。ここで、スパムメールを C_0 、通常メールを C_1 とラベル付けしよう。また、入力データを \mathbf{x} とする。このとき、クラス分類というのは、与えられた \mathbf{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。数値的な変数 $y = 0, 1$ を導入すると、 \mathbf{x} をクラス C_y へ分類するということは、 \mathbf{x} の所属クラスを表す離散ベクトル $y(\mathbf{x})$ の値を決めることであると言い換えられる。

$$\mathbf{x} \longrightarrow y(\mathbf{x}) \in \{0, 1\} \quad (3.1)$$

分類先のクラスが多数にわたる場合、多クラス分類とよび、分類先が C_1, C_2, \dots, C_K となり、ラベル $y(\mathbf{x})$ も 1 から K の整数値をとる。これは先ほどの例でいえば、電子メールを「仕事」「家族」「友人」「スパム」と複数のフォルダに分けることに対応する。

回帰

データに対するクラス y は離散的とは限らない。例えば、過去数日の気象データ \mathbf{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。このようにデータから、それに対応する実数値 (を並べたベクトル) \mathbf{y} を予測作業のことを回帰 (regression) という。つまり、回帰とは、与えられた \mathbf{x} を、対応する \mathbf{y} に変換するために関数 $\mathbf{y}(\mathbf{x})$ を決定する作業である。

$$\mathbf{x} \longrightarrow \mathbf{y}(\mathbf{x}) \in \mathbb{R} \quad (3.2)$$

回帰による応用的なタスクとして、言語を翻訳する機械翻訳や音声文字起こし音声認識、状況が通常と異なることを自動的に検知させる異常検知、データのサイズを圧縮させるデータ次元削減などがあげられる。

3.2 深層学習とは

3.3 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？データを科学的に分析する数理的手法といえば、統計学である。機械学習の手法もまた統計を基礎として構築される。そのような視点から特に統計的機械学習と呼ばれている。ここではまず、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指標を設計できるかについて説明する。

3.3.1 標本と推定

まず、データ (集合)(data(set)) やサンプル、標本 (sample) 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 (data point) の集まりからなる。手書き文字による画像認識の例でいえば、データ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 枚の画像がそれぞれデータ点とよばれる。ただしデータ点も略してデータとよばれる。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る。これらは用語の乱用であり、全く正しくない用法であるが、しばしば用いられているのが現状であるため、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。

このようなデータ点 (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。まず統計解析に用いるデータは、母集団から抽出 (sampling) されたものと

みなす。データの要素を 1 つ取り出すことは、正確には抜き取り (draw) という。そしてデータの分析から、データ自身ではなくその背後にある母集団についての知識を獲得することを目標とする。これは例えば疫学者が飲酒量と健康の関係を調べたいときに、地球上のすべての (=母集団) を調査する必要はないということと同じである。その代わりにランダムに選び取った (抽出した) 少人数に対する調査結果 (データ) を分析することで、人類すべてに通用する飲酒量と健康の相関関係を読み取ろうとするということである。

母集団の性質はデータ生成分布 (generative distribution) $P_{\text{data}}(\mathbf{x})$ により特徴づけられているものとする。つまり、不確実性を伴う現象を確率的にモデル化するということである。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。「サイコロを振ってどの目が出るか」という古典的な試行の例でも、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。そこで今後はどの場合でも、 \mathbf{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する。

$$\mathbf{x} \sim P_{\text{data}}(\mathbf{x}) \quad (3.3)$$

このよう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 $\mathbf{x} \sim P(\mathbf{x})$ は、ある変数 \mathbf{x} の実現値 \mathbf{x} が分布 $P(\mathbf{x})$ から生成している (生成された) ことを表す。

ここで抽出について 1 つ注意が必要である。我々は母集団について知りたいが、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。そこで基本的には無作為抽出されたデータのみを考えていくことにする。より正確には、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。

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 $P(\mathbf{x})$ 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 $P(\mathbf{x})$ をよく近似できると期待できるモデル分布 $P(\mathbf{x}; \boldsymbol{\theta})$ 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な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。

3.3.2 点推定

データがある未知の分布から生成していると考え、この確率分布のパラメータを決定しようとするのが総計的推定である。我々は母集団に直接アクセスすることはできないため、手持ちの有限要素のデータ集合 $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ からパラメータの尤もらしい値を計算するしかない。これ

を点推定という。点推定のため、データを定める確率変数 $\{x_1, x_2, \dots, x_N\}$ の関数である推定量

$$\hat{\theta}(x_1, x_2, \dots, x_N) \quad (3.4)$$

を作る。推定量のまた確率変数なので、データが具体的に与えられて、はじめて数値としてパラメータの推定値

$$\hat{\theta}^*(x_1, x_2, \dots, x_N) = \hat{\theta}(x_1, x_2, \dots, x_N) \quad (3.5)$$

を与えることになる。いずれにせよこの推定値は、いま考えているパラメータをよく近似するように作る。それによってデータの数値から、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かの指標がある。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

推定量のバイアスとは、推定量の期待値 $E[\hat{\theta}]$ と真の値 θ^* との差のことである。

$$b(\hat{\theta}) = E[\hat{\theta}] - \theta^* \quad (3.6)$$

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 $\lim_{N \rightarrow \infty} = 0$ 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。

$$\text{Var}(\theta) = E[(\hat{\theta} - \theta)^2] \quad (3.7)$$

一貫性

一貫性とは、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、 $N \rightarrow \infty$ に従い

$$\hat{\theta} \rightarrow \theta^* \quad (3.8)$$

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

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。

$$P(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.9)$$

μ と σ^2 の 2 つを決めればこの分布が定まるため、この 2 つがパラメータである。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を推定するにはそうすればよいか。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、その期待値は分布のパラメータ μ に一致している。

$$E_{\mathcal{N}}[x_n] = \int_{-\infty}^{\infty} x_n P(x_n) dx_n = \mu \quad (3.10)$$

これはもちろん分布に対する期待値だが、それを与えられたデータ $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ に関する平均値で近似してみる。つまり、本当の期待値であるガウス分布の期待値をサンプル平均で置き換えたものを μ の推定量 $\hat{\mu}$ とするのである。

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n \quad (3.11)$$

すると、これは見事に不偏推定量になっている。

$$\begin{aligned} E_{\mathcal{N}}[\hat{\mu}] &= \frac{1}{N} \sum_{i=1}^N E[x_n] \\ &= \mu \end{aligned} \quad (3.12)$$

では、 σ^2 はどうなっているか。

$$\begin{aligned} E_{\mathcal{N}}[(x_n - \mu)^2] &= \int_{-\infty}^{\infty} (x_n - \mu)^2 P(x_n) dx_n \\ &= \sigma^2 \end{aligned} \quad (3.13)$$

となる。したがって安直には、再びこの期待値をサンプル平均で近似すれば推定量 $\hat{\sigma}^2$ が得られると思う。

$$\hat{\sigma}^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 \quad (3.14)$$

これが不偏であるか見るために期待値を計算する。

$$\begin{aligned} E_{\mathcal{N}}[(x_n - \hat{\mu})^2] &= E_{\mathcal{N}}[(x_n - \mu)^2 - 2(x_n - \mu)(\hat{\mu} - \mu) + (\hat{\mu} - \mu)^2] \\ &= \sigma^2 - 2\frac{\sigma^2}{N} + \frac{\sigma^2}{N} \end{aligned} \quad (3.15)$$

$$\begin{aligned}
E_{\mathcal{N}} [\hat{\sigma}^2] &= E_{\mathcal{N}} \left[\frac{1}{N} \sum_{n=1}^N (x_n - \hat{\mu})^2 \right] \\
&= \frac{1}{N} E_{\mathcal{N}} \left[\sum_{n=1}^N \left\{ (x_n - \mu)^2 - 2(x_n - \mu)(\hat{\mu} - \mu) + (\hat{\mu} - \mu)^2 \right\} \right] \\
&= \frac{1}{N} \left\{ \sum_{n=1}^N E_{\mathcal{N}} [(x_n - \mu)^2] - 2 \sum_{n=1}^N E_{\mathcal{N}} [(x_n - \mu)(\hat{\mu} - \mu)] + N E_{\mathcal{N}} [(\hat{\mu} - \mu)^2] \right\} \\
&= \sigma^2 - \frac{2}{N^2} \sum_{n=1}^N \sum_{m=1}^N E_{\mathcal{N}} [(x_n - \mu)(x_m - \mu)] + \frac{1}{N^2} \sum_{n=1}^N \sum_{m=1}^N E_{\mathcal{N}} [(x_n - \mu)(x_m - \mu)] \\
&= \sigma^2 - \frac{1}{N^2} \sum_{n=1}^N E_{\mathcal{N}} [(x_n - \mu)^2] \\
&= \sigma^2 - \frac{1}{N} \sigma^2 \\
&= \left(1 - \frac{1}{N} \right) \sigma^2
\end{aligned}$$

となり、不偏ではない。ただし、 $N \rightarrow \infty$ 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。また、バイアスのない推定量は余分な係数を除いたものなので、

$$\hat{\hat{\sigma}}^2 = \frac{1}{N-1} \hat{\sigma}^2 = \frac{1}{N-1} \sum_{n=1}^N (x_n - \hat{\mu})^2 \quad (3.16)$$

と与えられる。

ベルヌーイ分布の場合

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のがベルヌーイ分布である。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、

$$P(x) = p^x (1-p)^{1-x} \quad (3.17)$$

と書ける。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、

$$E_P[x] = \sum_{x=0,1} x P(x) = 1 \cdot P(1) = p \quad (3.18)$$

$$\begin{aligned}
E_P[(x-p)^2] &= \sum_{x=0,1} (x-2px+p^2) P(x) \\
&= p^2 \cdots P(0) + (1-2p+p^2) \cdot P(1) \\
&= p(1-p)
\end{aligned} \quad (3.19)$$

と書くことができる。すると、パラメータの推定量は再びサンプル平均

$$\hat{p} = \frac{1}{N} \sum_{n=1}^N x_n \quad (3.20)$$

がよさそうである。実際,

$$E_P[\hat{p}] = \frac{1}{N} \sum_{n=1}^N E_P[x_n]$$

であり、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。

3.3.3 最尤推定

これまで、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はどうしようもない。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。こ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 $P_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ が与えられているとする。与えられたサンプル $\mathcal{D} = \{x_1, x_2, \dots, x_N\}$ 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なので、このデータ集合が得られる同時確率は

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

である。これを変数 $\boldsymbol{\theta}$ に対する量とみなして $L(\boldsymbol{\theta}) = P(x_1, x_2, \dots, x_N; \boldsymbol{\theta})$ と書き、尤度関数と呼ぶ。データ値が $\{x_1, x_2, \dots, x_N\}$ という観測値を取ったのは、パラメータ $\boldsymbol{\theta}$ の値が確率 $P(x_1, x_2, \dots, x_N; \boldsymbol{\theta})$ を大きくするようなものであったからであると考え。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 $\{x_1, x_2, \dots, x_N\}$ 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。

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argman}} L(\boldsymbol{\theta}) \quad (3.21)$$

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するため、実用上、対数尤度関数を最大化する。

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argman}} \ln L(\boldsymbol{\theta}) \quad (3.22)$$

このように何らかの目的関数を最大化、あるい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機械学習における常套手段である。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。機械学習では、最大値よりも最小値をして問題を書くことが多いので、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。

$$\boldsymbol{\theta}_{ML} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} (-\ln L(\boldsymbol{\theta})) \quad (3.23)$$

ガウス分布の場合

ガウス分布の例を考える． N 個のデータに対する尤度関数は

$$L(\boldsymbol{\theta}) = \prod_{n=1}^N \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_n - \mu)^2}{2\sigma^2}} \quad (3.24)$$

である．ただし， $\boldsymbol{\theta} = (\mu, \sigma^2)$ として 2 つのパラメータをまとめて表記した．対数尤度関数はもう少しシンプルで

$$\ln L(\boldsymbol{\theta}) = -\frac{N}{2} \ln \sigma^2 - \sum_n (x_n - \mu)^2 + \text{const.} \quad (3.25)$$

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，

$$\left. \frac{\partial \ln L(\boldsymbol{\theta})}{\partial \mu} \right|_{ML} = \frac{1}{\sigma_{ML}^2} \sum_{n=1}^N (x_n - \mu_{ML}) = 0 \quad (3.26)$$

$$\left. \frac{\partial \ln L(\boldsymbol{\theta})}{\partial \sigma^2} \right|_{\mu_L} = -\frac{N}{2\sigma_{ML}^2} + \frac{1}{2(\sigma_{ML}^2)^2} \sum_{n=1}^N (x_n - \mu_{ML})^2 = 0 \quad (3.27)$$

を解けばよい．これはすぐに解け，得られる推定量 $(\mu_{ML}, \sigma_{ML}^2)$ は先ほど見つけた推定量 $(\hat{\mu}, \hat{\sigma}^2)$ と一致する．

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

$$L(p) = \prod_{n=1}^N p^{x_n} (1-p)^{1-x_n} \quad (3.28)$$

となる．よって対数尤度は

$$\ln L(p) = \sum_{n=1}^N (x_n \ln p + (1-x_n) \ln (1-p)) \quad (3.29)$$

となる．これを解いて得られる最尤推定量 p_{ML} もまた，先ほど求めた推定量と一致している．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．

3.4 機械学習の基礎

機械学習ではデータ集合を学習アルゴリズムで処理することでコンピュータプログラムを学習させる．それによって，プログラムが与えられたタスクをよりうまくこなせるようにすることを目標にする．データ (集合) は機械学習の文脈では訓練データと呼ばれる．

学習を担うのは，人間がデザインする学習機械 (アーキテクチャ) である．アーキテクチャの学習プロセスを決めているのが学習アルゴリズムで，これもまた人間が設計する．アルゴリズムの詳細

細は学習させたいタスクやデータの種類によって変わる。ただし、一般的な設計思想は、まず与えられたタスクに対するパフォーマンスの良さを測る尺度を導入し、その値を改善するようなアルゴリズムを実現するというものである。この評価尺度を損失関数や誤差関数と呼ぶ。

3.4.1 教師あり学習

機械学習には教師あり学習や教師なし学習、強化学習など、いくつかの枠組が知られているが、深層学習の基礎になるものは教師あり学習である。教師あり学習の特徴は、用意すべき訓練データが必ず入力 \mathbf{x} と出力 \mathbf{y} のペアの形をとることである。つまり、訓練データは $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)\}$ という形をしている。機械学習の目的はこの入出力間の関係を推定することで、新しい未知の \mathbf{x} が与えられたときに対応する出力 \mathbf{y} を適切に予測できるようになることである。

$$\mathbf{x} \longrightarrow \mathbf{y} \quad (3.30)$$

統計学的な用語でいえば、説明変数 x と目標変数 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。

3.4.2 最小二乗法による線形回帰

入力 \mathbf{x} から出力 y を予測するのが回帰であった。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。そして、ここではパラメトリックなモデルを考える。つまり、常に y はある関数 $\hat{y} = f(\mathbf{x}; \mathbf{w})$ で表される規則で \mathbf{x} が決まっているとする。ただし、この関数のパラメータ \mathbf{w} 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が学習に相当する。また、データの観測には常にランダムなプロセスに起因する不確実な要素が介在するため、その効果をノイズ ϵ として、

$$y = f(\mathbf{x}; \mathbf{w}) + \epsilon \quad (3.31)$$

のようにモデル化する。ここで ϵ は確率変数で、ランダムなノイズによる規則性のかき乱しを表す。その一方で我々が知りたいのは \mathbf{x} と y の間の対応規則を捉える項 $f(\mathbf{x}; \mathbf{w})$ 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。

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{h}(\mathbf{x}) = \sum_{j=0} w_j h_j(\mathbf{x}) \quad (3.32)$$

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 $h_j(\mathbf{x})$ 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

項式を選んだ場合がいわゆる多項式回帰である。

$$f(x; \mathbf{w}) = \mathbf{w}^\top \mathbf{h}(x) = \sum_{j=0}^M w_j x^j \quad (3.33)$$

この多項式回帰も線形回帰の一種であることに注意。

回帰分析ではモデル $f(\mathbf{x}; \mathbf{w})$ 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 \mathbf{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、モデル $f(\mathbf{x}; \mathbf{w})$ をパフォーマンスの悪さを測る尺度である誤差関数を最小化することにより最適パラメータ \mathbf{w}^* を決定する。最も有名で理解しやすい誤差関数として、平均二乗誤差がある。

$$E_{\mathcal{D}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (\hat{y}(\mathbf{x}_n; \mathbf{w}) - y_n)^2 \quad (3.34)$$

$\hat{y}(\mathbf{x}_n; \mathbf{w})$ 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、 $(\hat{y}(\mathbf{x}_n; \mathbf{w}) - y_n)^2$ 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。平均二乗誤差を最小化する方法が、平均二乗法である。

探したいモデルは、できるだけデータへ当てはまりがよいものであるため、この誤差関数を最小化するようなパラメータ値

$$\mathbf{w}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}(E_{\mathcal{D}}(\mathbf{w})) \quad (3.35)$$

を \hat{y} のパラメータ値に選んだものが探していたモデルということになる。

ここで 1 つ概念的な注意を述べておく。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、ありとあらゆるデータに対してよい性能を示すモデルを作ることである。訓練データをもとにして学習機械を訓練し、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。したがって汎化を達成するために本来最小化すべき誤差関数は汎化誤差ということになる。

$$E_{\text{gen}}(\mathbf{w}) = E_{(\mathbf{x}, y) \sim P_{\text{data}}} [(\hat{y}(\mathbf{x}; \mathbf{w}) - y)^2] \quad (3.36)$$

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、この「本物」誤差を最小化することは不可能である。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もり、それを最小化している。式 (??) のようなサンプル平均で作った誤差を訓練誤差という。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、訓練誤差の最小化によって汎化に近づくことができる。

3.4.3 過適合と汎化

ここでは、汎化の実現がそう簡単ではないことを説明する。

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

$$\hat{y} = \sum_{j=0}^M w_j x^j \quad (3.37)$$

をデータにフィッティングする．この際，多項式の次数 M 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，学習されるパラメータを区別する．ハイパーパラメータ自体は学習アルゴリズムでは一切修正されない．この場合， M の重みパラメータの個数も決定しているため，まさにモデルの自由度を与えている．図 3.3 はフィッティングの結果である． $M = 1$ に対応する図 3.3(a) のように，データの豊かさ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．このような状況はアンダーフィッティングと呼ばれる．

そこで M をどんどん大きくしていくことで，データの複雑な構造をよりよく捉えられるようになるのではないかと考えられる．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) までを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．これが図 3.3(c) の状況である．すると，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．この状況は過学習と呼ばれる．

このようにモデルの自由度を減らしていくと過学習は下げられるがどんどん訓練誤差が大きくなり，逆に自由度を大きくしていくと訓練誤差をいくらでも小さくできる一方で過学習が起こって汎化誤差が増え始めてしまうトレードオフの状況になっている．すると，我々が見つけたい汎化の状況は 2 状態の間である図 (??)(b) にあると考えられる．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う状況である．したがって機械学習をうまくいかせるためには，まず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．

3.4.4 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．

$$E_{\text{new}}(\mathbf{w}) = E(\mathbf{w}) + \lambda R(\mathbf{w}) \quad (3.38)$$

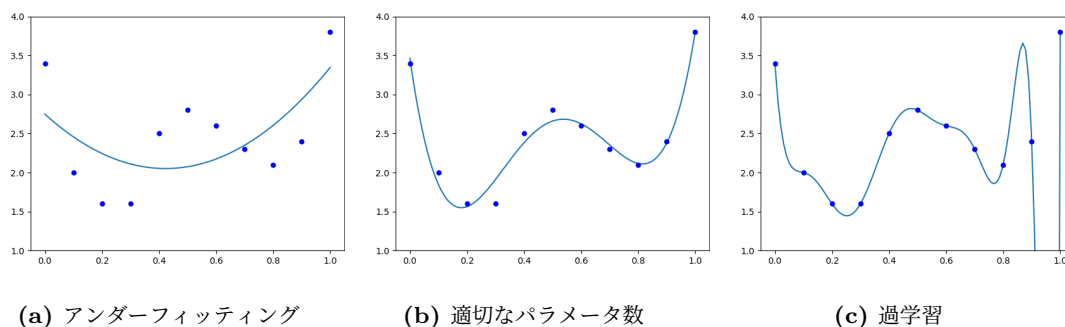


図 3.3: 1 次元データの線形回帰の例

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。

$$E_{\text{wd}}(\mathbf{w}) = E(\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w} \quad (3.39)$$

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 $\mathbf{w}^\top \mathbf{w}$ をできる限り小さくする解 \mathbf{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 \mathbf{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

$$E_{\text{wd}}(\mathbf{w}) = E(\mathbf{w}) + \lambda \sum_i |w_i| \quad (3.40)$$

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。

3.4.5 クラス分類

クラス分類の教師あり学習について説明する。クラス分類とは与えられた入力 \mathbf{x} を K 個のクラス $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K$ 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 つの入力が複数のクラスにまたがって所属することはないとする。また、入力には、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 y を用いればよい.

多クラス分類の場合

手書き数字 0 ~ 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. 最もシンプルなのは, K 個のクラスそれぞれに対応して $y = 1, 2, \dots, K$ 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

$$\text{bmt}(y) = (t(y)_1 \quad t(y)_2 \quad \cdots \quad t(y)_K)^\top \quad (3.41)$$

が 1-of-hot 符号化である. この手法では入力 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,

$$t(y = 1) = (1 \quad 0 \quad \cdots \quad 0)^\top \quad (3.42)$$

ということである. ここで, クロネッカーのデルタ記号

$$\delta_{i,j} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \quad (3.43)$$

を使うと, one-hot 表現ベクトルの成分は

$$t(y)_k = \delta_{y,k} \quad (3.44)$$

と書くことができる.

3.4.6 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.

$$\hat{y} = y(\mathbf{x}; \mathbf{w}) \quad (3.45)$$

例えば 0,1 の 2 値をとる目標変数に対し,

$$y(\mathbf{x}; \mathbf{w}) = f(\mathbf{w}^\top \mathbf{x} + b) \quad (3.46)$$

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 $P(\mathbf{x}, \mathbf{y})$ 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 $\mathbf{t}(\mathbf{y})^\top = (0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0)$ に対する $P(\mathbf{x}, \mathbf{y})$ は、入力 \mathbf{x} が所属クラスが \mathcal{C}_k である確率を意味する。この代わりに $P(\mathbf{x}, \mathcal{C}_k)$ 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、 \mathbf{x} について周辺化して $P(\mathbf{x})$ を求めてベイズの公式を使うことで条件付き確率 $P(\mathcal{C}_k|\mathbf{x})$ を求める。 $P(\mathcal{C}_k|\mathbf{x})$ がわかれば任意のデータ \mathbf{x} がわかれば任意のデータ \mathbf{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 \mathbf{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 $P(\mathcal{C}_k|\mathbf{x})$ を計算していた。このような回りくどいことをせずに $P(\mathcal{C}_k|\mathbf{x})$ を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。

3.4.7 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 $P(\mathcal{C}|\mathbf{x}) = P(\mathbf{x}|\mathcal{C}) = 1$ を満たす。ここでシグモイド関数

$$\sigma(u) = \frac{1}{1 + e^{-u}} = \frac{e^u}{1 + e^u} \quad (3.47)$$

を導入する。今考えていた、条件付き確率は

$$P(\mathcal{C}_1|\mathbf{x}) \quad (3.48)$$

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

$$u = \ln \frac{P(\mathcal{C}_1|\mathbf{x})}{1 - P(\mathcal{C}_1|\mathbf{x})} = \ln \frac{P(\mathcal{C}_1|\mathbf{x})}{P(\mathcal{C}_2|\mathbf{x})} \quad (3.49)$$

である。

$$e^u = \frac{P(\mathcal{C}_1|\mathbf{x})}{P(\mathcal{C}_2|\mathbf{x})} \quad (3.50)$$

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。

$$u = \mathbf{w}^\top \mathbf{x} + b \quad (3.51)$$

\mathbf{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、

$$P(y = 1|\mathbf{x}) = P(\mathcal{C}_1|\mathbf{x}), \quad P(y = 0|\mathbf{x}) = 1 - P(\mathcal{C}_1|\mathbf{x}), \quad (3.52)$$

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

$$P(y|\mathbf{x}) = (P(\mathcal{C}_1|\mathbf{x}))^y (1 - P(\mathcal{C}_1|\mathbf{x}))^{1-y} \quad (3.53)$$

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 $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ によって学習させるには、最尤法に従って、次の尤度関数

$$L(\mathbf{w}) = \prod_{n=1}^N (P(\mathcal{C}_1|\mathbf{x}_n))^{y_n} (1 - P(\mathcal{C}_1|\mathbf{x}_n))^{1-y_n} \quad (3.54)$$

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 \mathbf{w} は、対数オッズの部分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。

$$E(\mathbf{w}) = - \sum_{n=1}^N (y_n \ln P(\mathcal{C}_1|\mathbf{x}_n) + (1 - y_n) \ln (1 - P(\mathcal{C}_1|\mathbf{x}_n))) \quad (3.55)$$

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。

3.4.8 ソフトマックス関数

3.5 ニューラルネットワーク

3.5.1 神経細胞のネットワーク

人間の脳は、1000 億個以上ものニューロンと呼ばれる細胞が寄り集まって構成されている。この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。

ニューロンは、馴染みのある普通の細胞とは大きく異なる形状をしており、図??のようになっている。中央の膨らみは細胞体とよばれ、細胞核を担う中核となる部分であり、約 $10 \mu\text{m}$ ほどの大きさである。そして、ここから 2 種類の突起が伸びている。その 1 つは軸索とよばれる、1 本の細長くまっすぐな突起である。ヒトの場合、その全長は短くて 1mm、長いものだと 1m を越えるこ

ともある。長い軸索からはいくつもの分岐が出ており、これらを軸索側枝とよぶ。側枝はせいぜい数十 μm ほどの長さしかない。側枝の終端を軸索終末といい、重要な役割を果たす。

もう 1 つの突起が樹状突起である。樹状突起はまさに木の枝のように、幹である細胞体からたくさん伸びている。樹状突起はさらにいくつにも枝分かれし、枝というよりも植物の根のようにも見える。それらの全長はせいぜい数 mm であり、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、ある規則的な接合を繰り返して脳はできている。図??の丸で囲まれた部分に注目しよう。このシナプスとよばれる部位は、軸索から分岐した側枝の終端 (軸索終末) が他のシナプスと接合する部分である。軸索終末は主に他のニューロンの樹状突起や細胞体とシナプスを形成する。

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うところ、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。

3.5.2 形式ニューロン

深層学習の起源は、1943 年の W. マカロックと W. ピッツによる論文が源流と言われている。彼らはニューロンの活動が数理論理学的な手法でモデル化できることを発見した。それにより、神経活動の数理モデルと論理回路との対応が明らかになった。彼らは、形式ニューロンや人工ニューロンとよばれる素子を定義した。形式ニューロンは実際のニューロンの活動をモデル化したものである。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 $i = 1, 2, \dots$ から入力信号 x_i を受け入れる。図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物として、 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を表す重み w_i を導入して、層への

総入力を

$$u = \sum_i w_i x_i \quad (3.56)$$

と定義する． u は細胞体に入ってくる電気信号の総量に相当する．

この総入力を受けて、ニューロンは「軸索」方向へ出力を出す、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行列

$$\theta(x+b) = \begin{cases} 1 & (x \geq -b) \\ 0 & (x < -b) \end{cases} \quad (3.57)$$

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

$$z = \theta(u+b) = \theta\left(\sum_i w_i x_i + b\right) \quad (3.58)$$

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． z の値もまた、0 か 1 の 2 値で、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．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

NAND ゲートの出力を実現する形式ニューロンの回路を考えてみる．回路には、まず 2 つの入力値 x_1, x_2 を放出するだけの形式ニューロンが 2 つある．これらを入力ユニットと呼んでおく．入力ニューロンからの入力を受けて NAND と同じ出力値 $y(x_1, x_2)$ を出すニューロンを出力ニューロンと呼ぶことにする．このような回路全体を書いたのが図??である．

このような簡単なニューロン回路で NAND を再現するには、例えば

$$y = \theta(-x_1 - x_2 + 1.5) \quad (3.59)$$

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、これは論理回路そのものであることが確かめられる．

このようなニューロン回路を複雑に組み合わせることで、形式ニューロンを使ってどんな論理演算も実現できる．

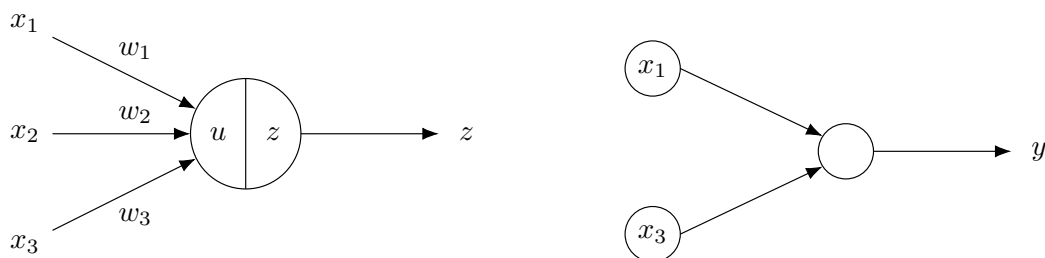


図 3.4: 形式ニューロン (左) と簡単なニューロン回路 (右)

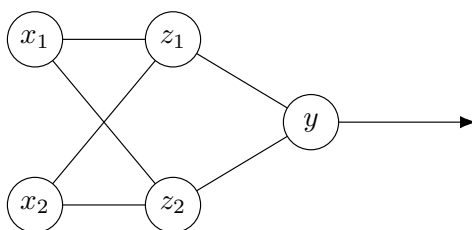


図 3.5: 2 層からなるパーセプトロン

3.5.3 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、どのようにニューロン回路を設計すれば、解きたい問題を計算してくれる効率的な回路が設計できるかはわからない。

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。

形式ニューロンを複数組み合わせることで作ったニューロン回路をパーセプトロンと呼ぶ。パーセプトロンは層構造をなしている。図 3.5 は 2 層構造をもつパーセプトロンの例である。ただし、入力層は層数には数えない。丸の中には、その形式ニューロンの出力値が書かれている。

一般に、複数の形式ニューロンの集まりからなる層の出力は、隣接する次の層の形式ニューロンへ入力する。各層からの出力が、また次層へと入力する構造を繰り返す。ただし、はじめの層は入力層と呼ばれ、入力値のベクトル $\mathbf{x}^T = (x_1, x_2, \dots)$ 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。つまり入力層の形式ニューロンの数は、入力ベクトルの次元の数だけある。

一方、最後の出力は出力層と呼ばれる。推定したい最終結果 $\mathbf{t}^T = (y_1, y_2, \dots)$ の各成分 y_i を出力値とする。それ以外の入出力間にある層は中間層や隠れ層と呼ばれる。各層のニューロン数については特に制限はない。図 3.5 の例では 2 つのニューロンからなる入力層と中間層、そして 1 つのニューロンからなる出力層を持っている。重みパラメータを使って学習ができるパーセプトロンは大きな注目を集め、1960 年代の第 1 次ニューラルネットのブームを巻き起こした。

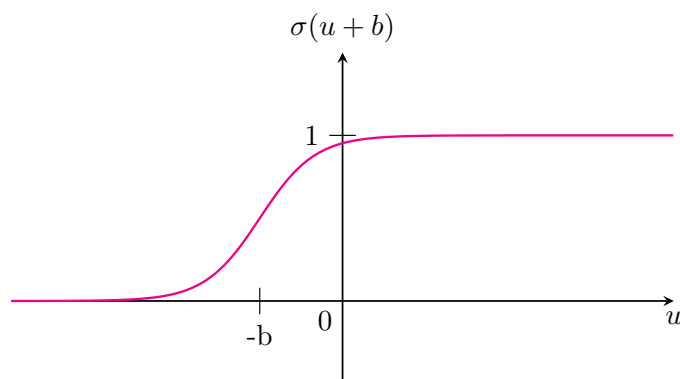


図 3.6: シグモイド関数

3.5.4 順伝播型ニューラルネットワーク

ニューラルネットはユニット (unit) と呼ばれる素子からなる。図 3.7(左) に示したユニットは、パーセプトロンにおける形式ニューロンに他ならない。ただし各ユニットは実数値の入出力をもつ。つまり各ユニットの出力値として、どんな実数を考えてもよいということである。もう 1 つの大きな違いは、活性化関数として微分可能増大関数を採用するということである。例えばシグモイド関数がよく用いられていた。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

$$u = \sum_i w_i x_i \quad (3.60)$$

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 (activation function) 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。

$$z = f(u + b) = f\left(\sum_i w_i x_i + b\right) \quad (3.61)$$

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

$$\sigma(u + b) = \frac{1}{1 + e^{-u-b}} \quad (3.62)$$

である。図 3.6 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。

順伝播ニューラルネット (feedforward neural network) 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。これを略してニューラルネットとも呼ぶ。図 3.8 はその典型的な一例である。 l が層数で、 $l = 1$ が入力層、 $l = 4$ が出力層、そして、 $l = 2, 3$ が中間層、または隠れ層である。信号の伝達は $l = 1, 2, 3, 4$ の順に進み、そのため順伝播型と呼ばれる。

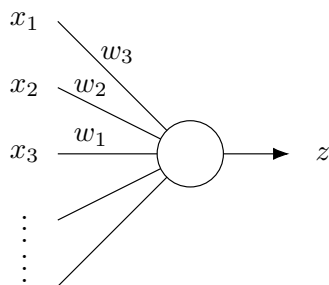


図 3.7: ユニットの構造. x_1, x_2, \dots の入力を受け, z を出力する

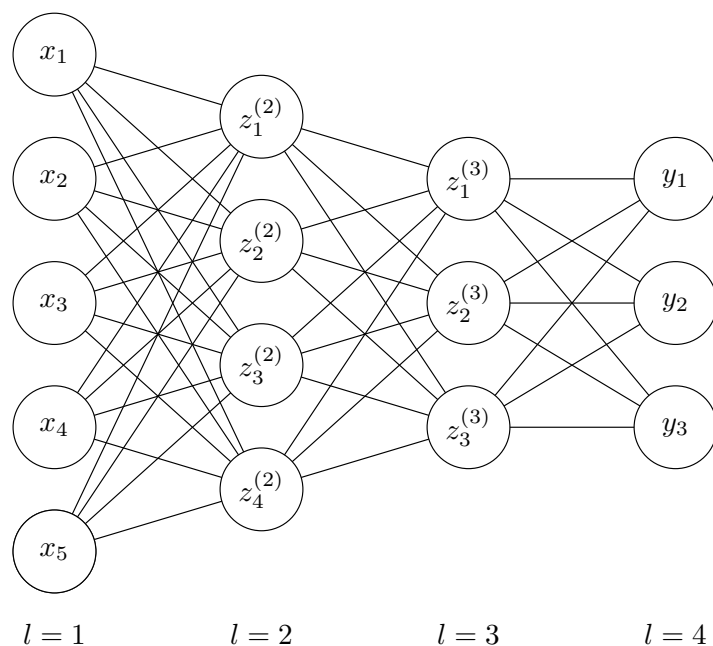


図 3.8: 入力層と 3 層からなる順伝播ニューラルネットワークの例. ユニットを表す丸の中に変数 z は, 各ユニットの出力値を意味する.

まず, 入力層は 5 つのユニットからなっている. 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 $\mathbf{x}^\top = (x_1, x_2, x_3, x_4, x_5)$ の各成分を出力する. したがって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. 入力層はニューラルネットワークの層数には数えない.

入力層は入力ベクトル \mathbf{x} の次元数だけユニットをもつ. また, 「第 1 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.

$$\mathbf{z}^{(1)} = \mathbf{x} \quad (3.63)$$

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

$$u_j^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)} \quad (3.64)$$

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

$$z_j^{(l)} = f^{(l)} \left(u_j^{(l)} + b_j^{(l)} \right) = f^{(l)} \left(\sum_i w_{ji}^{(l)} z_i^{(l-1)} + b_j^{(l)} \right) \quad (3.65)$$

で与えられる。活性化関数は $f^{(l)}$ はユニットごとに変わってもかまわないが、一般的には層ごとに共通の関数を用いる。

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 $\mathbf{u}^{(l)} = (u_j^{(l)})$ で、出力をまとめて $\mathbf{z}^{(l)} = (z_j^{(l)})$ 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

$$\mathbf{W}^{(l)} = \begin{pmatrix} w_{11}^{(l)} & w_{12}^{(l)} & \cdots \\ w_{21}^{(l)} & w_{22}^{(l)} & \\ \vdots & & \ddots \end{pmatrix} \quad (3.66)$$

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

$$\mathbf{u}^{(l)} = \mathbf{W}^{(l)} \mathbf{z}^{(l-1)}, \quad \mathbf{z}^{(l)} = f^{(l)} \left(\mathbf{u}^{(l)} + \mathbf{b}^{(l)} \right) \quad (3.67)$$

とまとめることができる。ただし、一般のベクトル $\mathbf{v}^\top = (v_1 v_2 v_3 \cdots)$ 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。

$$f(\mathbf{v}) = f \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} (f(\mathbf{v}))_1 \\ (f(\mathbf{v}))_2 \\ (f(\mathbf{v}))_3 \\ \vdots \end{pmatrix} \equiv \begin{pmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ \vdots \end{pmatrix} \quad (3.68)$$

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。そしてそれを $i = 0$ というラベルで呼ぶことにする。

$$z_0^{(l)} = 1 \quad (3.69)$$

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、

$$\sum_{i=0} w_{ji}^l z_i^{l-1} = \sum_{i=1} w_{ji}^l z_i^{l-1} + w_{j0}^l = \sum_{i=1} w_{ji}^l z_i^{l-1} + b_j^l \quad (3.70)$$

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。

$$\mathbf{u}^{(l)} = \mathbf{W}^{(l)} \mathbf{z}^{(l-1)}, \quad \mathbf{z}^{(l)} = f^{(l)}(\mathbf{u}^{(l)}) \quad (3.71)$$

最後に出力層について考える． $L - 1$ 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 $\mathbf{z}^{(L-1)}$ を入力され，それを変換することでニューラルネットワークの最終出力 $\mathbf{y} = \mathbf{z}^{(L)}$ 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， $\mathbf{h} = \mathbf{z}^{(L-1)}$ が入力 \mathbf{x} の表現であり，この表現 \mathbf{h} の回帰分析を通じて \mathbf{x} と \mathbf{y} の関係を推定するのが出力層の役割である．

$$\hat{\mathbf{y}} = \mathbf{z}^{(L)} = f^{(L)}(\mathbf{u}^{(L)}), \quad \mathbf{u}^{(L)} = \mathbf{W}^{(L)} \mathbf{h} \quad (3.72)$$

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ワークは，各総出の処理を順次入力ベクトル \mathbf{x} に施していくことで， \mathbf{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．

$$\hat{\mathbf{y}} = f^{(L)}\left(\mathbf{W}^{(L)} f^{(L-1)}\left(\mathbf{W}^{(L-1)} f^{(L-2)}\left(\dots \mathbf{W}^{(2)} f^{(1)}(\mathbf{x})\right)\right)\right) \quad (3.73)$$

3.6 ニューラルネットワークによる機械学習

順伝播ニューラルネットワークは入力 \mathbf{x} を受け取ると出力

$$\mathbf{y}(\mathbf{x}; \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L)}) \quad (3.74)$$

を出す．出力地は重み $\mathbf{W}^{(l)}$ やバイアス $\mathbf{b}^{(l)}$ の値を変えることで変化する．これらのパラメータをまとめて $\boldsymbol{\theta}$ と書くことにする．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 $\boldsymbol{\theta}$ 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 $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1, \dots, N}$ を用意する．そのうえで訓練データ \mathbf{x}_n を入れた際のニューラルネットワークの出力値 $\mathbf{y}(\mathbf{x}_n; \boldsymbol{\theta})$ と，目標値 \mathbf{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 $E(\boldsymbol{\theta})$ 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

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} E(\boldsymbol{\theta}) \quad (3.75)$$

が学習に対応する

式 (3.74) のように考えると，入力から一気に \mathbf{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 \mathbf{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．するともっとも最後の第 $L - 1$ 層からの出力は，入力 \mathbf{x} に対する深層表現 \mathbf{h} である．

$$\mathbf{h}(\mathbf{x}; \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}) = \mathbf{z}^{(L-1)}(\mathbf{x}; \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}) \quad (3.76)$$

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

もし表現 \mathbf{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。

$$\mathbf{y} = \mathbf{W}^{(L)} \mathbf{h} \quad (3.77)$$

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。

$$\mathbf{y}(\mathbf{x}; \boldsymbol{\theta}) = f^{(L)} \left(\mathbf{W}^{(L)} \mathbf{h} \right) \quad (3.78)$$

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、 $\mathbf{y}(\mathbf{x}; \boldsymbol{\theta})$ 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。

$$E(\boldsymbol{\theta}) = \frac{1}{2} \sum_{n=1}^N (\mathbf{y}(\mathbf{x}_n; \boldsymbol{\theta}) - \mathbf{y}_n)^2, \quad \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} E(\boldsymbol{\theta}) \quad (3.79)$$

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 $\mathbf{y}(\mathbf{x}; \boldsymbol{\theta}^*)$ は、入力 \mathbf{x} の値に応じて \mathbf{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

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 $\mathbf{h} = \mathbf{z}^{(L-1)}$ 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 $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1, \dots, N}$ 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 $\hat{y} = P(y = 1|\mathbf{x})$ 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 $E_{P(y|\mathbf{x})}[y|\mathbf{x}] = \sum_{y=0,1} yP(y|\mathbf{x}) = P(y = 1|\mathbf{x})$ 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 $P(y = 1|\mathbf{x})$ 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。

$$y(\mathbf{x}; \boldsymbol{\theta}) = P(y = 1|\mathbf{x}; \boldsymbol{\theta}) = \sigma \left(\sum_i w_i^{(L)} h_i \right) \quad (3.80)$$

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、ユニットへの対数オッズである。

$$f^{(L)} = \sigma, \quad u^{(L)}(\mathbf{x}; \boldsymbol{\theta}) = \ln \frac{P(y = 1 | \mathbf{x}; \boldsymbol{\theta})}{1 - P(y = 1 | \mathbf{x}; \boldsymbol{\theta})} \quad (3.81)$$

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 \mathbf{x} を入力した際の出力値から、 \mathbf{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、 $P(y = 1 | \mathbf{x})$ 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 $P(y = 1 | \mathbf{x}; \boldsymbol{\theta})$ を推定するベルヌーイ分布

$$P(y | \mathbf{x}; \boldsymbol{\theta}) = P(y = 1 | \mathbf{x}; \boldsymbol{\theta})^y (1 - P(y = 1 | \mathbf{x}; \boldsymbol{\theta}))^{(1-y)} \quad (3.82)$$

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 $y(\mathbf{x}_n; \boldsymbol{\theta}) = P(y = 1 | \mathbf{x}_n; \boldsymbol{\theta})$ に対し、

$$E(\boldsymbol{\theta}) = - \sum_{n=1}^N (y_n \ln y(\mathbf{x}_n; \boldsymbol{\theta}) + (1 - y_n) \ln 1 - y(\mathbf{x}_n; \boldsymbol{\theta})) \quad (3.83)$$

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けた深層表現を表現学習できることになる。

多クラス分類

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 $\mathbf{h} = \mathbf{z}^{(L-1)}$ 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 $y = 1, 2, \dots, K$ 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 \mathbf{x} が $y = k$ 番目のクラスに属する確率 $P(y = k | \mathbf{x})$ 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。

$$y_k(\mathbf{x}; \boldsymbol{\theta}) = P(y = k | \mathbf{x}; \boldsymbol{\theta}) = \text{softmax}_k \left(u_1^{(L)}, \dots, u_K^{(L)} \right) \quad (3.84)$$

$$\mathbf{u}^{(L)} = \mathbf{W}^{(L)} \mathbf{h} \quad (3.85)$$

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 \mathbf{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。

$$y \longleftrightarrow \mathbf{t}(y) = (t(y)_j) = (0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0)^\top \quad (3.86)$$

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。

$$\mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n)\}_{n=1, \dots, N} \quad (3.87)$$

このベクトル \mathbf{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

$$E(\boldsymbol{\theta}) = - \sum_{n=1}^N \sum_{k=1}^K t_{n,k} \ln y_k(\mathbf{x}_n; \boldsymbol{\theta}) \quad (3.88)$$

であり、これが誤差関数に他ならない。

3.6.1 活性化関数

3.7 畳み込みニューラルネットワーク

3.8 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。

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} L(\boldsymbol{\theta}) \quad (3.89)$$

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。

3.8.1 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。

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 $\boldsymbol{\theta}^{(0)}$ 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 $t = 0, 1, 2, \dots$ 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

$$\nabla E(\boldsymbol{\theta}) = \frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \equiv \left(\frac{\partial E(\boldsymbol{\theta})}{\partial \theta_1}, \dots, \frac{\partial E(\boldsymbol{\theta})}{\partial \theta_D} \right) \quad (3.90)$$

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 $\theta_1, \theta_2, \dots, \theta_D$ でラベル付けした。 D はニューラルネットワークの全パ

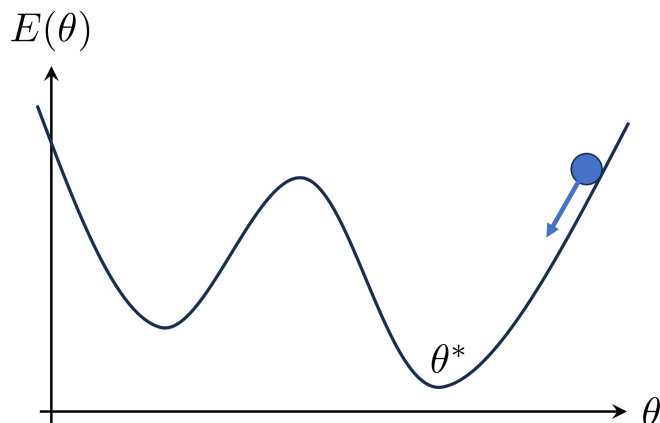


図 3.9: 勾配降下法による極小値のを見つけ方

ラメータ数である．すると時刻 t で位置 $\theta^{(t)}$ 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．

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}, \quad \Delta\theta^{(t)} = -\eta \nabla E(\theta^{(t)}) \quad (3.91)$$

右辺で次時刻の $\theta^{(t+1)}$ を定義する操作を $t = 0, 1, 2, \dots$ と順次繰り返していくことで，どんどん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 $\Delta\theta^{(t)}$ の大きさを決めるハイパーパラメータ η は学習率 (learning rate) 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．

3.8.2 局所的極小値の問題

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っ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 つでも見つければ，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。

3.8.3 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つ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 $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1,2,\dots,N}$ が与えられたとき、誤差関数は各学習サンプル要素 $(\mathbf{x}_n, \mathbf{y}_n)$ で計算した誤差の和として表現できた。

$$E(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N E_n(\boldsymbol{\theta}) \quad (3.92)$$

例えば平均二乗誤差を用いるならば

$$E_n(\boldsymbol{\theta}) = \frac{1}{2} (\mathbf{y}(\mathbf{x}_n; \boldsymbol{\theta}) - \mathbf{y}_n)^2 \quad (3.93)$$

であり、 K クラス分類ならば交差エントロピー

$$E_n(\boldsymbol{\theta}) = - \sum_{k=1}^K t_{nk} \log y_k(\mathbf{x}_n; \boldsymbol{\theta}) \quad (3.94)$$

を用いた。先ほどの勾配降下法では式 (3.92) 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

$$E^{(t)}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}^{(t)}|} \sum_{n \in \mathcal{B}^{(t)}} E_n(\boldsymbol{\theta}) \quad (3.95)$$

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。

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}, \quad \Delta\theta^{(t)} = -\varepsilon \nabla E^{(t)}(\theta^{(t)}) \quad (3.96)$$

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 $E^{(t)}(\theta)$ の形もランダムに変化する。したがって、ずっと同じ関数 $E(\theta)$ 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 $\nabla E_n(\theta)$ 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。

3.8.4 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。

3.9 改良された勾配降下法

3.9.1 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 3.10(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 3.10(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 3.10(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。

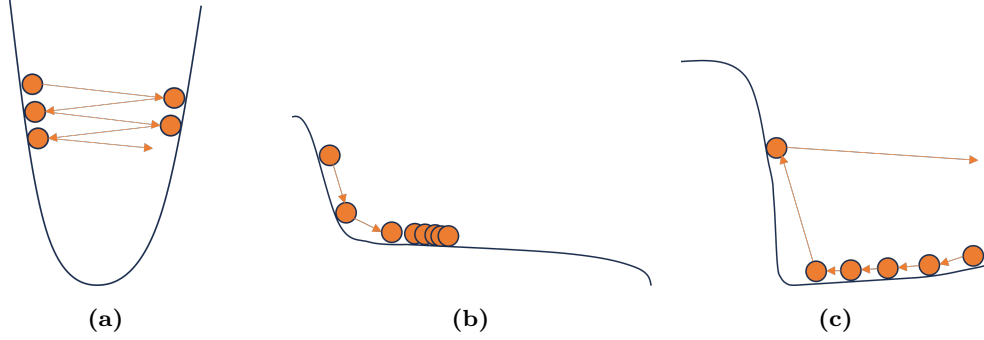


図 3.10: (a) 谷で振動. (b) プラトーで停止. (c) 絶壁で反射

3.9.2 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 $\Delta\theta^{(t-1)}$ が負の値で、現在の勾配 $\nabla E(\theta^{(t)})$ 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 3.11 のように今回の更新量 $\Delta\theta^{(t)}$ 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。

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)} \quad (3.97)$$

$$\Delta\theta^{(t)} = \mu\Delta\theta^{(t-1)} - (1 - \mu)\eta\nabla E(\theta^{(t)}) \quad (3.98)$$

初期値は $\Delta\theta^{(0)} = 0$ 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 $\nabla E(\theta^{(t)})$ が一定の領域を考える。この場合、式 (??) の終端速度は $\Delta\theta^{(t)} = \Delta\theta^{(t+1)} = \Delta\theta$ として更新式を解くことで、

$$\Delta\theta = -\eta\nabla E(\theta) \quad (3.99)$$

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。

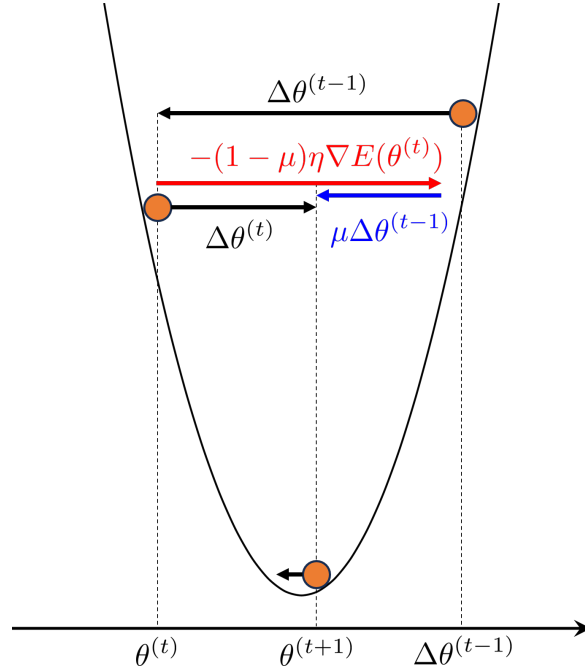


図 3.11: モーメンタム法による谷での振動の抑制

3.9.3 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えば、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。

$$\Delta \theta_i^{(t)} = - \frac{\eta}{\sqrt{\sum_{s=1}^t (\nabla E(\theta^{(s)})_i)^2}} \nabla E(\theta^{(t)}) \quad (3.100)$$

左辺は $\Delta \theta^{(t)}$ の第 i 成分であり、 $\nabla E(\theta^{(t)})_i$ 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 の欠点としては，学習の初期に勾配が大きいとすぐさま更新量 $\Delta\theta_i^{(t)}$ が小さくなってしまふことである．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．

3.9.4 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つた．これは過去のすべての勾配の情報を収集してしまつていたためである．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．

$$v_{i,t} = \rho v_{i,t-1} + (1 - \rho)(\nabla E(\theta^{(t)})_i)^2 \quad (3.101)$$

$$\Delta\theta_i^{(t)} = -\frac{\eta}{\sqrt{v_{i,t} + \epsilon}} \nabla E(\theta^{(t)})_i \quad (3.102)$$

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

$$\text{RMS}[\nabla E_i]_t = \sqrt{v_{i,t} + \epsilon} \quad (3.103)$$

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．

3.9.5 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 θ 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 $\Delta\theta$ 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．

$$[\Delta\theta] = L, \quad [\nabla E(\theta)] = L^{-1} \quad (3.104)$$

3.9.6 Adam

しかし、勾配法では、この次元性を合わない両者を学習率で比例させてしまっている。この mismatches が学習率に押し付けられてしまうために、適切な学習率のスケールが問題に応じてさまざまに変化してしまうのである。

ロバストな学習率を実現させるためには、この mismatches を解消しなくてはならない。そこで

3.10 誤差逆伝播法

第 4 章

機械学習によるイジング模型の相転移検出

- 4.1 配位データの生成方法
- 4.2 相の分類器による相転移検出
- 4.3 温度測定器による相転移検出
- 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

謝辞を書く.

参考文献

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

付録 A

aaa

aaa

A.1 bbb

bbb