第 I 部

理論

第 1 章

数学的準備

ここでは本稿の目的である手書き数字の認識を行うニューラルネットワークの構築に必要な数学的事項を説明する。大まかに分けて、線形代数、確率論、情報理論の知識が必要になる。

1.1 線形代数

1.1.1 スカラー、ベクトル、行列、テンソル

スカラー（scalar）は $-1.4,\ 1,\ \pi$ などの単一の数のことをいう。本稿の数式上では $a, b, c, \ldots$ のようにアルファベットのイタリック体で書き表すことにする。

スカラーを 1 次元状に並べたものをベクトル（vector）という。本稿ではベクトルを $\boldsymbol{x}$ のように、ボールドイタリック体の小文字のアルファベットで表す。ベクトルの成分は $x_1,\ y_i$ のようにスカラーに要素番号を添えることで表す。

2 次元の数の配列を行列（matrix, pl. matrices）という。本稿では A などのイタリック体の大文字で行列を表し、A_{ij} のように 2 つの添字を付すことで成分を表すことにする。

3 次元以上の数の配列をテンソル（tensor）といい、本稿では A のようにサンセリフ体で表記する。テンソル A の (i, j, k) 成分は A_{ijk} のように表す。

1.1.2 積の計算

行列 A について、その成分 A_{ij} と A_{ji} を入れ替えた行列を転置行列（transposed matrix）または単に転置（transpose）といい、$A^\top$ と表記する。すなわち $A_{ij} = \left(A^\top\right)_{ji}$. ベクトルは 1 つの列ベクトルからなる行列とみなせば、ベクトルの転置は行ベクトルである。転置の例を次に示す。

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix} \quad \longrightarrow \quad A^\top = \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \end{bmatrix},$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \longrightarrow \quad \boldsymbol{x}^\top = \begin{bmatrix} x_1, x_2, \ldots, x_n \end{bmatrix}.$$

スカラー a については、それを 1×1 の行列とみなせば転置をとっても変わらない。すなわち $a^\top = a$ である。

$m \times n$ の行列 A と $n \times p$ の行列 B に対して、行列積（matrix product）$C = AB$ が次のように定義できる。

$$C_{ij} = \sum_k A_{ik} B_{kj}. \tag{1.1}$$

この行列 C のサイズは $m \times p$ になる。

2 つの同じサイズのベクトル $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$ と $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$ について、ドット積（dot product）は行列積 $\boldsymbol{x}^\top \boldsymbol{y}$ となり、これはスカラーである。

$$\boldsymbol{x}^\top \boldsymbol{y} = \begin{bmatrix} x_1, \ldots, x_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = x_1 y_1 + \cdots + x_n y_n = \sum_{k=1}^{n} x_k y_k. \tag{1.2}$$

$\boldsymbol{x}^\top \boldsymbol{y}$ はスカラーであるから、$\left(\boldsymbol{x}^\top \boldsymbol{y}\right)^\top = \boldsymbol{x}^\top \boldsymbol{y}$ である。また式 (1.2) から分かるように、$\boldsymbol{x}^\top \boldsymbol{y} = \boldsymbol{y}^\top \boldsymbol{x}$ である。ゆえに $\boldsymbol{x}^\top \boldsymbol{y} = \left(\boldsymbol{x}^\top \boldsymbol{y}\right)^\top = \boldsymbol{y}^\top \boldsymbol{x}$ である。

行列のアダマール積（Hadamard product）を計算することがある。同じサイズの行列 A と B について、そのアダマール積 $A \odot B$ は、それぞれの行列の成分ごとの積を計算した行列である。すなわち $(A \odot B)_{ij} = A_{ij} \cdot B_{ij}$ である。この定義はより一般にテンソルにも拡張できる。

数式の見た目をシンプルにするために、アインシュタインの縮約記法（Einstein summation convention）を使うことがある。テンソルの成分についての計算時に、同じ項の中で同じ添字が現れた場合、総和の記号 $\sum$ を省略して書くのだ。この記法を用いれば、式 (1.1) は $C_{ij} = A_{ik} B_{kj}$，式 (1.2) は $\boldsymbol{x}^\top \boldsymbol{y} = x_k y_k$ と書ける。アインシュタインの縮約記法を用いるときは必ずその旨を表すようにする。

1.1.3　行列の微分

行列 A に対して、その A_{ij} の成分を偏微分演算子 $\partial / \partial A_{ij}$ に置き換えた行列を $\boldsymbol{\nabla}_A$ と書く。もちろんこれは演算子であるから、何らかの関数に左から作用させたときに実体を持つ。

$$(\boldsymbol{\nabla}_A f)_{ij} = \frac{\partial f}{\partial A_{ij}}.$$

1.1.4　ノルム

ベクトル $\boldsymbol{x} = [x_1, \ldots, x_n]^\top$ の L^p ノルム（L^p norm）を

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

と定義する。

L^2 ノルムは数学的に扱いやすく、よく用いられるため、ユークリッドノルム（Euclidean norm）ともよばれる。特に L^2 ノルムの 2 乗は、そのベクトル自身とのドット積に一致する。

$$\|\boldsymbol{x}\|_2^2 = x_1^2 + \cdots + x_n^2 = \boldsymbol{x}^\top \boldsymbol{x}.$$

1.2　確率論

1.2.1　確率論を使うことの意義

コンピュータ科学ではほとんどの場合、完全に決定論的で、確実な対象を扱うが、機械学習では不確実・非決定論的な量を扱わなければならない。不確実性を生み出す要因は 3 つある。1 つめは扱う対象そのものが確率性を持つ場合（例：量子力学や完璧にシャッフルされたカードなど）、2 つめは扱う対象の振る舞いを決める変数を観測できない・しない場合（例：モンティ・ホール問題において回答者の回答が当たりかハズレかは決定論的に決まっているが、ドアを開けるまで結果は不確実である）、最後は観測した情報を破棄した場合（例：物体の位置を離散化したとき、正確な位置は不確実となる）である。

確実だが複雑な規則よりも、不確実だが単純な規則を用いるほうが実用的であるため、確率論が用いられる。

頻度確率（frequentist probability）とは、反復可能な試行を無限回くりかえしたときに、ある事象が現れる頻度という意味での確率のことである。いっぽう反復可能な試行でなくとも、信念の度合い（degree of belief）、可能性の大きさという意味での確率をベイズ確率（Bayesian probability）という。確率論ではこれら 2 つの意味での確率を、同等に扱う。

確率論はある命題のもっともらしさが与えられたときに、対象の命題が真であるもっともらしさを決定するための、不確実性を扱うための論理学の拡張とみなすことができる。

1.2.2　確率変数と確率分布

確率変数（random variable）とは起こりうる事象に対応した、ランダムな値を取れる変数のことである。確率変数を x のようにローマン体の小文字で書き、x がとりうる値は x のようにイタリック体で書く。確率変数がベクトル値のときはそれらは太字で $\mathbf{x}$, $\boldsymbol{x}$ のように書く。確率変数は離散値でも連続値でもよい。たとえばサイコロを振ったときの「6 の目が出る」という事象は $\mathrm{x} = 6$ に対応させる、コイントスをしたときの「裏が出る」「表が出る」という事象をそれぞれ $\mathrm{x} = 0,\ \mathrm{x} = 1$ に対応させる。このように確率変数は、事象の集合（標本空間）から数の集合への関数と考えることもできる。

確率質量関数（probability mass function, PMF）とは、離散型確率変数に対してその値をとるときの確率を対応させた関数のことであり、大文字 P で書き表す。たとえば確率 p で表（$\mathrm{x} = 1$）、確率 $1 - p$ で裏（$\mathrm{x} = 0$）が出るようなコインでコイントスをするとき、確率質量関数は

$$P(\mathrm{x} = x) = \begin{cases} p, & (x = 1) \\ 1 - p & (x = 0) \end{cases}$$

とかける（これはベルヌーイ分布とよばれる）。このコイントスを n 回行ったうち k 回だけ表が出る確率は

$$P(\mathrm{y} = k) = \binom{n}{k} p^k (1-p)^{n-k} = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}$$

となる（これは二項分布とよばれる）。このとき「確率変数 y は母数 (n, p) の二項分布 $\mathrm{B}(n, p)$ に従う」といい、これを $\mathrm{y} \sim \mathrm{B}(n, p)$ と表記する。

$\mathrm{x} = x$ かつ $\mathrm{y} = y$ であるときの確率 $P(\mathrm{x} = x, \mathrm{y} = y) = P(x, y)$ は同時確率分布とよばる。

関数 P が離散型確率変数 x の確率質量関数であるためには以下の性質を満たさなければならない。

- 定義域は確率変数のとりうる値すべての集合である。
- その集合を A とすると、$\forall x \in A$ に対して $0 \leq P(x) \leq 1$.
- 正規化されている（normalized）: $\sum_{\forall x \in A} P(x) = 1$.

たとえば、確率変数 x が k 個の異なる離散値 $x_1, x_2, \ldots, x_k$ をとるとき、確率質量関数を

$$P(\mathrm{x} = x_i) = \frac{1}{k}$$

とすれば一様分布を定義することできる。すべての i で和をとれば

$$\sum_{i=1}^{k} P(\mathrm{x} = x_i) = \frac{1}{k} \sum_{i=1}^{k} 1 = \frac{1}{k} \cdot k = 1$$

となって、正規化されていることがわかる。

離散型確率変数に対して確率質量関数とよんでいたものを、連続型確率変数に対しては確率密度関数（probability density function, PDF）とよぶ。確率密度関数 p は以下の性質を満たさなければならない。

- 定義域は確率変数のとりうる値すべての集合である。
- その集合を I とすると、$\forall x \in I$ に対して $0 \leq p(x)$. ただし $p(x) \leq 1$ である必要はない。
- $\int_I p(x)\,\mathrm{d}x = 1$.

$\mathrm{x} = x$ であるときの確率は $p(x)$ と直接得られるわけではなく、代わりに x が微小区間 $[x, x + \delta x]$ の値をとるときの確率が $p(x)\delta x$ で与えられる。区間 $[a, b]$ に x が存在する確率は $\int_{[a,b]} p(x)\,\mathrm{d}x$ で求められる。

確率密度関数の例として、一様分布がある。

$$u(x; a, b) = \begin{cases} \dfrac{1}{b-a}, & x \in [a, b], \\ 0, & x \notin [a, b]. \end{cases}$$

積分すれば 1 になる。x が区間 $[a, b]$ において一様分布にしたがうことを、$\mathrm{x} \sim U(a, b)$ と表す。

1.2.3　条件付き確率と確率の連鎖律

ある事象が起こったもとでの別の事象が起こる確率を条件付き確率（conditional probability）という。$\mathrm{x} = x$ が与えられたもとでの $\mathrm{y} = y$ が起こる条件付き確率 $P(\mathrm{y} = y \mid \mathrm{x} = x) = P(y \mid x)$ は

$$P(\mathrm{y} = y \mid \mathrm{x} = x) = \frac{P(\mathrm{y} = y, \mathrm{x} = x)}{P(\mathrm{x} = x)} \tag{1.3}$$

と定義される。

多数の確率変数 $\mathrm{x}_1, \ldots, \mathrm{x}_n$ に対する同時確率分布は式 (1.3) を用いれば、1 つの確率変数についての条件付き確率に分解してそれらの積として表現できる。このことを確率の連鎖律（chain rule of probability）という。

$$P(\mathrm{x}_1, \ldots, \mathrm{x}_n) = P(\mathrm{x}_1) \prod_{i=2}^{n} P(\mathrm{x}_i \mid \mathrm{x}_1, \ldots, \mathrm{x}_{i-1}).$$

1.2.4　ベイズの定理

条件付き確率 $P(y \mid x)$ を知っているときに、別の条件付き確率 $P(x \mid y)$ を求めたい状況はよくある。確率分布 $P(x)$ が分かっていれば、これは次のベイズの定理（Bayes' theorem）を用いることで計算できる。

$$P(x \mid y) = \frac{P(x)P(y \mid x)}{P(y)} = \frac{P(x)P(y \mid x)}{\sum_x P(x)P(y \mid x)}. \tag{1.4}$$

1.2.5　期待値と分散

確率変数 x が確率分布 P に従うとき、関数 $f(x)$ の平均を期待値（expectation）といい、記号 $\mathbb{E}_{x\sim P}[f(x)]$ と表す。ただし確率変数 x が確率分布 P に従うことが明らかな場合は、単に $\mathbb{E}[f(x)]$ と書く。離散型確率変数についての期待値の定義は

$$\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x)$$

であり、連続型確率変数の期待値は積分を用いて

$$\mathbb{E}_{x\sim p}[f(x)] = \int p(x)f(x)\,\mathrm{d}x$$

と定義される。

また関数 $f(x)$ のばらつき度合いを表すのに分散（variance）が用いられる。

$$\mathrm{Var}[f(x)] = \mathbb{E}\Big[\big(f(x) - \mathbb{E}[f(x)]\big)^2\Big] = \mathbb{E}\Big[\big(f(x)\big)^2\Big] - \big(\mathbb{E}[f(x)]\big)^2.$$

1.3　情報理論

1.3.1　自己情報量とシャノン・エントロピー

「地球は自転している」という情報よりも「明日地球に隕石が衝突する」という情報の方が意外性が高く、より大きな価値をもつ気がする。このような直感的な「情報の価値」のようなものを数学的に表現したい。具体的には

- 起こりやすい事象の情報量は少なく、確実に起こる事象の情報量はないとする
- 起こるのが珍しい事象ほど情報量は大きい
- 独立な事象については情報量は足し算で表される。例えば「コイントスを 2 回行ったところ表が 2 回出た」という事象の情報は、「コイントスを 1 回行ったところ表が出た」という事象のそれよりも、2 倍の大きさをもつ

というようなものを数式で表現したい。この 3 つの性質を満たすためには次のような量を作ればよい。

$$I(x) = -\log P(x).$$

これを事象 $\mathrm{x} = x$ の自己情報量（self-information）という。本稿では log は常に自然対数を表すものとし、その場合の自己情報量の単位はナット（nats）という。

自己情報量は 1 つの事象のみについての情報量であるが、全体の情報量の平均をシャノン・エントロピー（Shannon entropy）という。

$$H(\mathrm{x}) = H(P) = \mathbb{E}_{\mathrm{x}\sim P}[I(x)] = -\sum_i P(x_i)\log P(x_i).$$

たとえば確率 p で表が出るコインでコイントスをするときのシャノン・エントロピーは $H(P) = -p\log p - (1-p)\log(1-p)$ となる。この関数を図示すると図 1.1 のようになり、$p = 0, 1$（結果が確実にわかっている）ときに $H(P) = 0$, $p = 1/2$（結果は不確実）のときに最大値を取ることがわかる。一般に離散型確率変数に対する確率分布のシャノン・エントロピーが最大になるのは一様分布のときである。

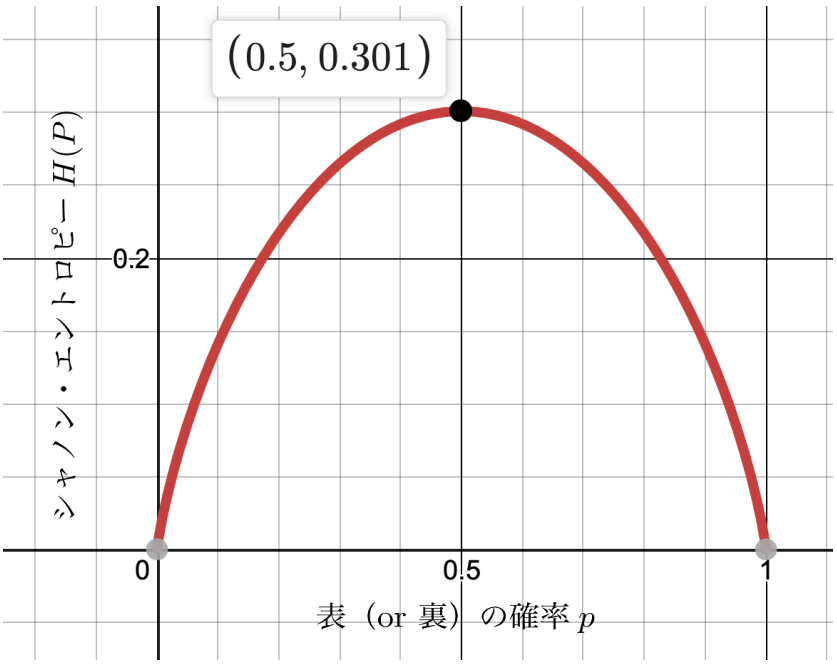

図 1.1　二値確率変数のシャノン・エントロピー

1.3.2　KL ダイバージェンスと交差エントロピー

同じ確率変数 x に対して異なる確率分布 $P(\mathrm{x})$ と $Q(\mathrm{x})$ があったとき、それらがどれほど異なるのかを表す量として KL ダイバージェンス（Kullback–Leibler divergence）がある。

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log\frac{P(x)}{Q(x)}\right] = \sum_i P(x_i)\log\frac{P(x_i)}{Q(x_i)}.$$

KL ダイバージェンスは非負の量であり、等号成立は P と Q が同じときである。よってこれは P と Q の距離のような概念と考えられるが、一般に $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$ であるため距離の公理の 1 つ $d(x,y) = d(y,x)$ を満たさず、距離と呼ぶのは正しくない。

また次の量を交差エントロピー（cross-entropy）という。

$$H(P,Q) = H(P) + D_{\mathrm{KL}}(P\|Q).$$

これを少し式変形をすると

$$\begin{aligned} H(P,Q) &= H(P) + D_{\mathrm{KL}}(P\|Q) \\ &= -\sum_i P(x_i)\log P(x_i) + \sum_i P(x_i)\log\frac{P(x_i)}{Q(x_i)} \\ &= -\sum_i P(x_i)\log Q(x_i) \end{aligned}$$

となり、$\mathrm{x} \sim P$ のもとでの $Q(x)$ のシャノン・エントロピー $\mathbb{E}_{\mathrm{x}\sim P}[-\log Q(x)]$ を表しているとわかる。KL ダイバージェンスと比較して取り除かれている部分 $\sum_i P(x_i)\log P(x_i)$ は Q に依存しないため、交差エントロピーを Q に関して最小化することは KL ダイバージェンスを最小化することと等価である。交差エントロピーはのちにコスト関数としてよく用いる。

1.4　機械学習の基礎

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.1　学習アルゴリズム

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.2　線形回帰

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.3　ベイズ統計

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

1.4.3.1　最尤推定

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.3.2　最大事後確率推定

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.4　過学習の問題

1.4.4.1　過学習・過小学習

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.4.2　交差検証

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.4.3　正則化

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu.

1.4.5　最適化問題

1.4.5.1　ラグランジュの未定乗数法と KKT 法

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. 関数 $J(\boldsymbol{\theta})$ はパラメータのベクトル $\boldsymbol{\theta}$ によって特徴付けられているものとする。

1.4.5.2　バッチ勾配降下法

関数 $J(\boldsymbol{\theta})$ をバッチ勾配降下法（batch gradient descent, BGD）を用いて最小化することがここでの目的である。

まず初期値 $\boldsymbol{\theta}_0$ をランダムに選び、その点での勾配 $\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$ を計算する。そして

$$J(\boldsymbol{\theta}_1) \leftarrow J(\boldsymbol{\theta}_0) - \boldsymbol{\eta}^\top \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_0)$$

のように関数 $J(\boldsymbol{\theta})$ を更新する。さらにその点 $\boldsymbol{\theta}_1$ を初期値として同じ操作を反復すれば、最小点 $\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$ に収束する。係数のベクトル $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_k]^\top$ の成分あるいはベクトル自体のことを、ステップ幅あるいは機械学習の文脈では学習率（learning rate）とよばれる。

数値計算上では、関数の微分はその定義の近似を用いればよい。

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}.$$

学習率の値は大きすぎず小さすぎないのがよい。もし学習率が大きすぎると、ステップごとに最小点を通り過ぎてしまい、収束が遅くなるか収束できなくなるからである。反対に小さすぎると 1 ステップが小さいため収束が遅くなる。

関数 $J(\boldsymbol{\theta})$ が m 個の訓練データから得られる平均二乗誤差 $\mathrm{MSE}(\boldsymbol{\theta})$ であることはよくある。その場合、勾配降下法はコストが高い方法である。訓練データの数を m, パラメータの数を k とすれば、平均二乗誤差は m 項からなる。パラメータのそれぞれの成分についての微分を計算するので、1 ステップあたり $m \times k$ 回の計算が必要である。ステップは 1000 回行うことが普通であるので、例えば $m = 10000$, $k = 10$ であるとすれば全体で 10^8 回の計算が必要になってしまい、データ数が大きいと遅いアルゴリズムであることがわかる。

1.4.5.3　確率的勾配降下法

バッチ勾配降下法ではコスト関数は各データに関する項の和の形

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{m} J_i(\boldsymbol{\theta})$$

をしていた。このニュアンスを強調して上のアルゴリズムをバッチ勾配降下法（batch gradient descent）とよぶ。それに対して訓練データからランダムに 1 つ選んで、そのコスト関数のみについての勾配を計算して更新してゆく手法を確率的勾配降下法（stochastic gradient descent）という。訓練データはステップごとに選び直す。

確率的な性質を持つため、最小点に向かってゆく仮定は複雑になる。平均的には最小値に向かって緩やかに小さくなっていくのだが、最小値周りに達すると勾配は様々な方向を向いているので、1 箇所に落ち着くことがない。そのため確率的勾配降下法による最終的なパラメータは十分よいものだが最適ではない。

しかしコスト関数が複数の極小値を持つような複雑な関数の場合、確率的な性質のおかげで、関数の谷となっている部分から抜け出し、真の最小値にたどり着く可能性は大きくなる。

まとめると確率的勾配降下法は、その無作為性によって局所的な最小値から逃れる可能性はあがるが、最小値に落ち着かないというジレンマをもつ。その問題を解決するために、勾配の大きさに応じて学習率を変化させる方法がある。勾配が大きければ大きく前進し、勾配がゼロに近くなったら慎重に前進するのだ。金属が結晶化する過程がヒントになっていて、このアルゴリズムを焼きなまし法（simulated annealing）とよぶ。

1.4.5.4　ミニバッチ勾配降下法

ミニバッチ勾配降下法（minibatch gradient descent）はバッチ勾配降下法と確率的勾配降下法の中間的なアルゴリズムである。ステップごとに訓練データから複数個のデータをランダムに抜き出して、それらのコスト関数の勾配から最小値を探し出す。ランダムに選ばれた訓練データの集合をミニバッチとよ

ぶ。N ステップ目の、m' 個の訓練データからなるミニバッチを $\mathbb{B}_N$ と表せば、N ステップ目のコスト関数は

$$J_N(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}^{(i)} \in \mathbb{B}_N} J_i(\boldsymbol{\theta}) = \frac{1}{m'} \sum_{\boldsymbol{x}^{(i)} \in \mathbb{B}_N} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

とかける。

1.5　フィードフォーワード・ニューラルネットワーク

1.6　畳み込みニューラルネットワーク

1.6.1　畳み込み処理

実数を引数にとる実関数 $I(t)$ と重みの関数 $K(t)$ を用いて

$$S(t) = \int I(\tau) K(t - \tau) \, \mathrm{d}\tau$$

という関数 $S(t)$ を生成する。この処理を畳み込み（convolution）とよび、$S(t) = (I * K)(t)$ のように表記する。これらの関数の引数が離散値であるならば、離散畳み込みを

$$S(t) = (I * K)(t) = \sum_{\tau} I(\tau) K(t - \tau)$$

と定義する。畳み込みネットワークの文脈では I を入力（input）、K をカーネル（kernel）またはフィルター（filter）、S を特徴マップ（feature map）とよばれる。

2 変数に対する離散畳み込みは

$$S(i, j) = \sum_{m} \sum_{n} I(i - m, j - n) K(m, n)$$

となる。ただし畳み込みの可換性 $(I * K)(i, j) = (K * I)(i, j)$ が用いられている。一方で、ニューラルネットワークの実装においては相互相関（cross-correlation）とよばれる次の量も畳み込みとよび、よく用いられている。

$$S(i, j) = \sum_{m} \sum_{n} I(i + m, j + n) K(m, n).$$

1.6.2　2 次元畳み込み処理の例

2 次元のデータに畳み込み処理を行う例を図 1.2 に示す。簡単のため入力 I はサイズが 6×6 の行列であるとし、その成分の値をグレースケールで視覚化してある。2 次元のフィルター K は 3×3 の行列である。フィルターはその中心を決められるように、行数と列数ともに奇数であると便利である。まず図 1.2 中の赤色の四角形で示したように、入力 I の左上の位置にフィルター K の成分が全て含まれるように重ねる。同じ位置にある成分同士を掛け合わせ、総和をとると 4 になるので、その値が特徴マップ S の左上の成分になる。次に図 1.2 中の緑色の四角形で示したように、重ね合わせていたフィルター K を右に 1 つずらして同様の計算をする。入力 I の右端までこの操作を繰り返したら 1 行下がる。この操作を入力 I の右下に達するまで繰り返す。なお、入力 I に重ねているフィルター K の影のようなこの領域を、神経科学の言葉になぞらえて、受容野（receptive field）とよぶ。

より一般に、2 次元の入力 I のサイズが $N \times M$, 2 次元のフィルター K のサイズが $(2k+l) \times (2l+1)$ であるとする。ただし N, M, k, l は 0 を含まない自然数である。(i, j) 成分の値をそれぞれ $I(i, j)$, $K(i, j)$ としたとき、2 次元畳み込み処理によって得られる値 $S(i, j)$ は

$$S(i,j) = \sum_{n=0}^{2l} \sum_{m=0}^{2k} I(i+m, j+n) K(1+m, 1+n)$$

となる。i と j の範囲は $i = 1, \ldots, M-2k$, $j = 1, \ldots, N-2l$ で、特徴マップ S のサイズは $(M-2k) \times (N-2l)$ となる。ただし Python では配列の要素番号は 0 から始まるので、その形式に合わせるには総和の下端を $n = 0$ から $n = -1$ のように書き換えればよい。

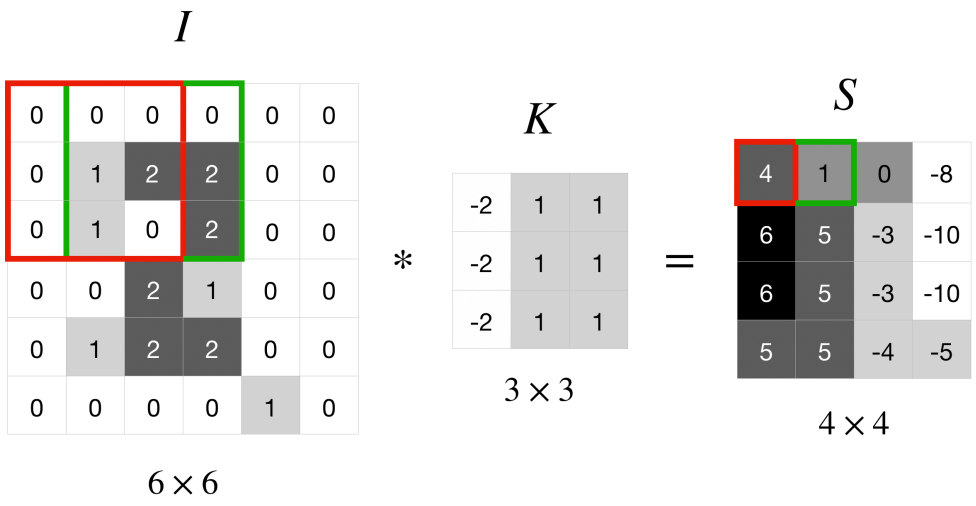

図 1.2　2 次元畳み込み処理の例

1.6.3　ゼロパディングとストライド

1.6.2 の例では、入力のサイズが 6×6 であったのに対し、出力のサイズは 4×4 に縮小している。これでは畳み込みを行うごとに出力が縮小していってしまう。このことを防ぐには、図 1.3 のようにあらかじめ入力の周囲にゼロを追加すればよい。この「余白」のように追加された部分をゼロパディング（zero padding）とよぶ。図 1.3 の例ではゼロパディングの幅は 1 にしているが、この幅はフィルター K のサイズによって変化させればよい。そのサイズが $(2k+l) \times (2l+1)$ ならば、入力 I の左右に幅 k, 上下に幅 l のゼロパディングをそれぞれ追加する。

また、受容野をずらす間隔のことをストライド（stride）という。図 1.2 と図 1.3 の例ではストライドは 1 であった。ストライドを大きくすると出力は小さくなる。

ゼロパディングの幅やストライドの値は、ライブラリで畳み込みネットワークを使用する際、引数として渡す。

1.6.4　フィルターの効果

フィルターを使うと、入力のフィルターに近い部分が強調されて特徴マップとして得られる。

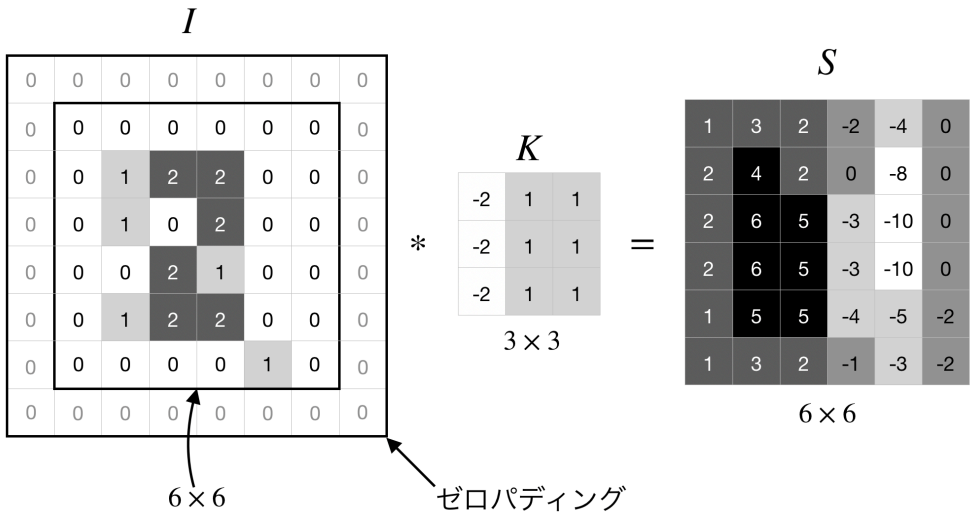

図 1.3　2 次元畳み込み処理の例

1.2 と図 1.3 の例におけるフィルター

$$K = \begin{bmatrix} -2 & 1 & 1 \\ -2 & 1 & 1 \\ -2 & 1 & 1 \end{bmatrix} \tag{1.5}$$

について考える。入力 I において、左側で小さい値を持ち、右側で大きい値を持つような領域をこの受容野が通ったとき、特徴量マップの対応するピクセルは大きな値を持つが、反対に左側で値が大きく右側で小さい領域を受容野が通ったときは、特徴量の値は小さくなる。左右の方向に均一な入力の領域では、このフィルターを使うと 0 に近い値を返す。このように、式 (1.5) のフィルターは入力の縦線の右側のエッジを強調させる効果をもつ。なおこのフィルターは全要素の総和が 0 になるように設計した。これにより、強調させたい入力の特徴を 0 以上として定めることができる。

同様に様々なフィルターを考案することができる。例えば

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

はそれぞれ横線、点、右肩上がりの斜め線を強調するようなフィルターとなる。

畳み込みニューラルネットワークでは、フィルターを重みとして訓練する。

このように、フィルターは入力のある特徴を誇張させる効果がある。畳み込みニューラルネットワークは訓練によってタスクに役立つフィルターを発見し、それらの組み合わせを学習していく。