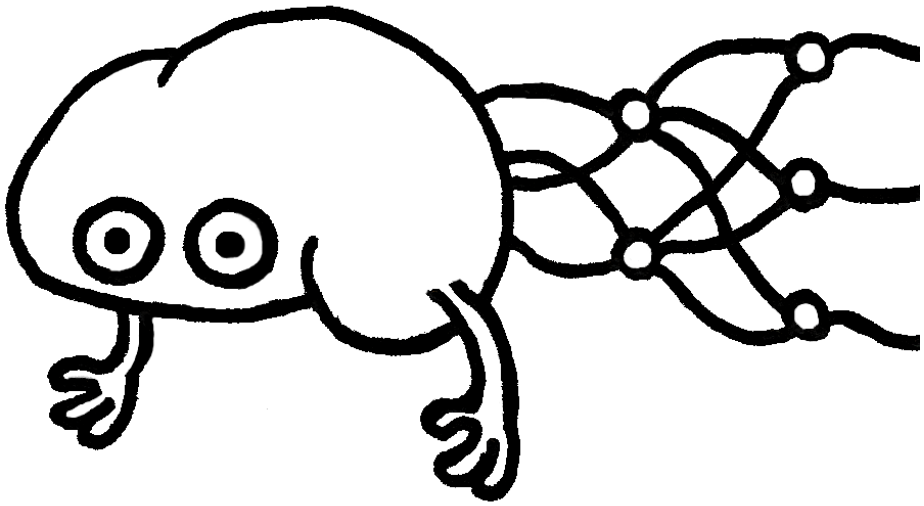
京都大学人工知能研究会

KaiRA

Kyoto univ. AI Research Association



2019 NF vol.3

京都大学人工知能研究会　Kaira

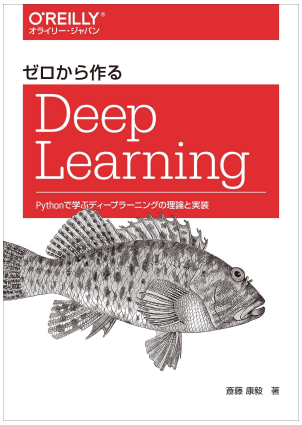
活動内容

1. 本の輪読
   深層学習や機械学習に関する専門書籍の輪読を行います。各章に担当者を設定し、プレゼンやハンズオンなど様々な形態で理解を深めます。

2. ソフトウェア開発
   学んだことを活かして各自が興味あるプロジェクトに取り組みます。成果は勉強会や学園祭で発表します。

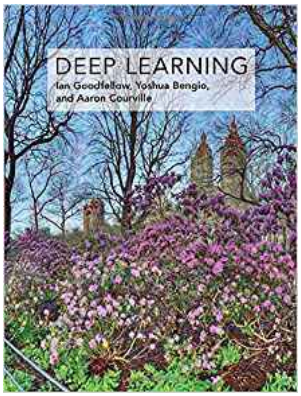
サークル情報

代表　大山百々勢（工学部情報学科 2 回）
活動日　毎週木曜日　18:30
場所　京大病院内
会費　無料
入会資格　誰でも
Mail　kyoto.kaira@gmail.com
Website　https://kyoto-kaira.github.io/contact.html
Twitter　@kyoto-kaira



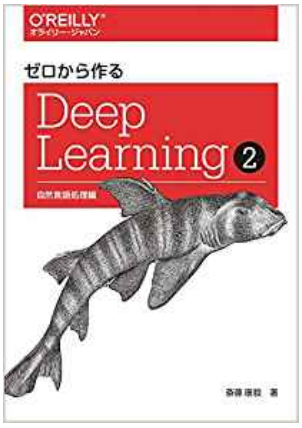
(a) www.amazon.co.jp/dp/4873117585

(b) www.amazon.co.jp/dp/4839962510



(c) www.amazon.co.jp/dp/0262035618

(d) www.amazon.co.jp/dp/4065172519



(e) www.amazon.co.jp/dp/4873118360

図 1 今までに輪読してきた本

はじめに

こんにちは！

近年、深層学習を用いた技術は画像や音声の認識やある条件下での最適な行動の学習 (ゲームや車の自動運転など) において素晴らしい成果を発揮しています。こうした技術は今後私たちの生活にとってより一層欠かせないものになると期待されます。

私たち KaiRA は理系文系・専攻・回生・大学を問わず深層学習や機械学習 (人工知能) に興味のある学生が一丸となって活動しています。毎週一回一緒に集まり、専門書籍の輪読会を行ったり面白いと感じた話題をみんなで共有したりしています。輪読会では、発表者のスライドや質疑応答を通じて理解を深めることができます。

今回の NF では、これまでの勉強会の内容を簡単にまとめたポスターや、実際に作成したデモを展示しております。この冊子ではディープラーニングについて簡単に紹介し、関連する興味深い話題を紹介します。私たちの展示を通じて機械学習や深層学習の面白さが伝われば幸いです。

京都大学人工知能研究会
KaiRA 会長
大山百々勢

目次

第 1 章

深層学習入門

1.1　はじめに

現在、自分は人工知能とは無縁であると言える人はほとんどいないのではないだろうか。なぜなら現在あらゆる分野で人工知能の技術が用いられているからである。例えばネットフリックスで映画をおすすめされることがあるが、これは視聴履歴からユーザーがどのような映画を好むのかを人工知能が学習し、それに合わせておすすめをしているのである。また技術面だけでなく映画やゲームのテーマとして人工知能が取り上げられることが増えてきた。今年の仮面ライダーでは人工知能を搭載したアンドロイドが社会で活躍している様子が描かれている。

この様に人工知能が急激に注目を浴びるようになってきているが、その中でも特に期待されているのは深層学習の分野であろう。「深層学習」「ディープラーニング」という言葉に耳なじみのある方も多いだろう。しかし、原理を知っている人となるとかなり数は減る。実際、ニュースで取り上げられることがあっても原理が説明されることはほとんどない。ここでは深層学習の原理について説明したいと思う。「ディープ」とは何が「ディープ」なのか、「ラーニング」とは何をもって「ラーニング」なのかわかっていただければ幸いです。

1.2　深層学習の位置付け

まず、深層学習という言葉の位置付けをする。深層学習は機械学習、人工知能と次のような関係となっている。

深層学習 ⊂ 機械学習 ⊂ 人工知能

一つずつ説明していこう。まず、人工知能というのは自分で学習や思考をする人工物の

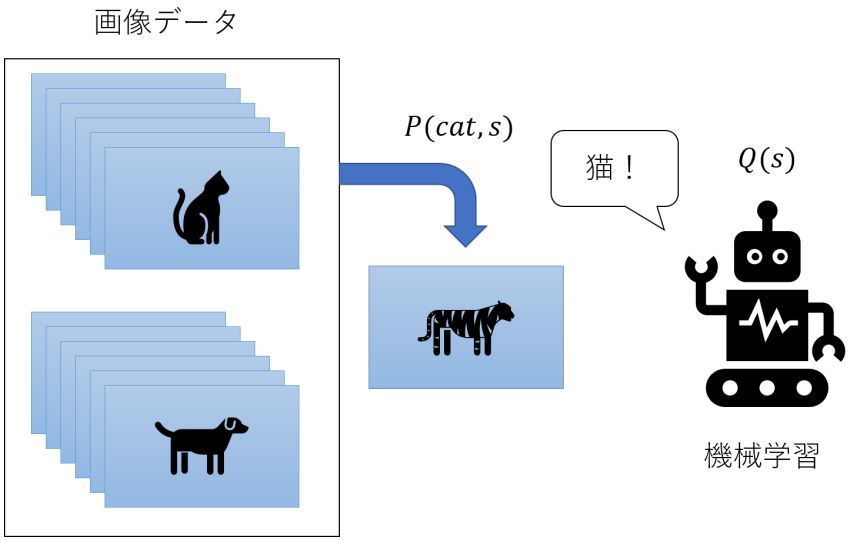


図 1.1 犬猫判別機

総称である。ここでの「学習」や「思考」というのは厳密な定義があるわけではない。人工知能というのは以下で説明する機械学習や深層学習を緩く包み込むような概念である。例えば、「コロ助は人工知能である」ということができる。

次に、機械学習というのはデータから情報を抽出したりなどの特定の作業をするコンピュータアルゴリズムまたは統計的なモデルである。例えば、excel で近似曲線が引かれることがあるがああいった線形回帰などが機械学習の一例である。機械学習にはある特定の目的が想定されるため、何のためのからくりなのかわからないコロ助は機械学習には属さない。

よくある機械学習のタイプとしてデータ分布を学習するものが多い。例えば、ある画像を入力したときその画像に写っているのが猫なのか犬なのか判別する機械学習を考える（図 1.1）。画像はデジタルであれば高々有限個のピクセルの集まりなので各ピクセルの数値を成分としたベクトル $\boldsymbol{s}$ を考えることができる。また画像はそれが猫か犬かを表すラベルが張られていたとする。するとデータは

$$(\boldsymbol{s_1}, cat), (\boldsymbol{s_2}, dog), \cdots, (\boldsymbol{s_N}, dog)$$

のように表される。このようなデータを多く用意した状況を考えると確率分布 $P(\boldsymbol{s}, a)$ が考えられる。但し、a は cat または dog。この時、入力された画像 $\boldsymbol{s_i}$ が猫であるかどうかを予測することを言い換えれば

$$Q(\boldsymbol{s_i}) \simeq P(cat|\boldsymbol{s_i}) \tag{1.1}$$

となるような関数 $Q(\boldsymbol{s})$ を求めることと等価である。一度そのような関数を得てしまえば、入力画像 $\boldsymbol{s}$ が与えられたとき、猫の画像である確率 $Q(\boldsymbol{s})$ が高ければ猫と出力、低ければ犬と出力すればよい。以下では入力 x に対し、y を出力するようなタイプの機械学習を考える。

そして機械学習の中でも「深い層構造」を持った「ニューラルネットワーク」を機構として含むものを深層学習という。「ニューラルネットワーク」は日本語でいえば神経回路であり、脳の中の細胞間のつながりの構造を想像してもらえればよい。そして細胞が層構造を作り情報が順番に層に伝わっていく。その層の数がとても多いことを「深い」と形容しているのである。もう少し「ニューラルネットワーク」と「深い層構造」という言葉についての説明を補足する。

1.3 ニューラルネットワーク

前節でニューラルネットワークは脳の中の細胞のネットワークのようなものだといったがそもそも細胞同士の情報の伝達がどのようにされているのかミクロな視点に立って考える。

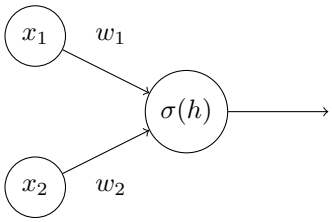
脳の細胞のことをニューロンというがニューロン間はシナプスで繋がっている。ニューロン同士の情報のやり取りはシナプスの「重み付け」とニューロンの「発火」によって特徴づけられる。まず、「重み付け」はニューロンのつながりの強さみたいなもので、それに応じて電気信号が定数倍されて伝えられる。但し、必ず電気信号を受け取ったニューロンが次のニューロンへと信号と伝えるとは限らず、ある一定以上の信号の強さを受信しなければ次に送信しない。基準を超えた信号を受け取り、次に送信したときニューロンが「発火」したという。このようなふるまいを数理的なモデルで表すと入力を x_1, x_2、出力を y としたとき次のようになる。

$$y = \max(w_1 x_1 + w_2 x_2 + b, 0) \tag{1.2}$$

またグラフで表したものが図（1.2）である。但し、max 関数は次のように定義

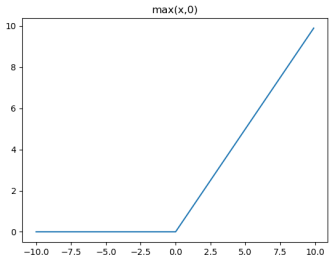
$$\max(x, y) = \begin{cases} x & (x > y) \\ y & (y > x) \end{cases} \tag{1.3}$$

この時 w_i を**重み**、b を**バイアス**または**閾値**という。max 関数により非線形となっているがこのことが層構造を入れたときに重要な意味を持つ。また今回は max 関数を用いたが深層学習では他に tanh 関数や sigmoid 関数を用いることがある。但し、どちらも非線形関数であり、入力の値が小さければ出力の値は 0 に近づく（図 1.3）。

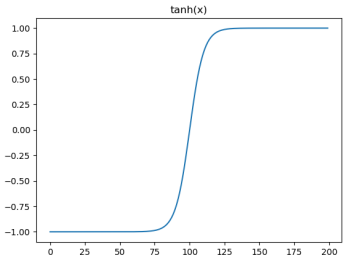


但し、$h = w_1x_1 + w_2x_2 + b$
σ は非線形関数

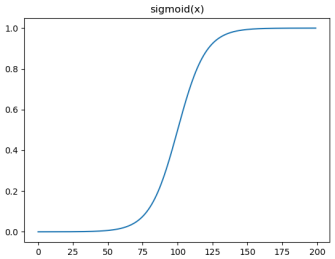
図 1.2 ニューロンとシナプスの数理モデル



(a) max 関数



(b) tanh 関数



(c) sigmoid 関数

図 1.3 活性化関数の例

この様に二つの細胞間の情報の伝わり方が決まってしまったら後は複数の細胞をどう繋げるかである。入力用の細胞と出力用の細胞を決めてやればある関数が出来上がることがわかる。但し、余りにも無秩序な構造だと扱いにくい。というのも実際に機械学習として用いる場合はシナプスの重みとバイアスをうまく調整してやる必要がある。整った構造で作ってやればどのように調整すればいいのかがわかりやすくなる。そこで深層学習では「層構造」を導入するのである。

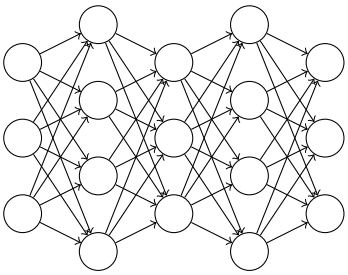


図 1.4 ディープニューラルネットワーク

1.4 層構造

細胞を並べて「層構造」を導入する。以下細胞のことを**ノード (node)** という。まず、一列にノードを並べたものを用意する。但し、このノード間にはシナプスがなく情報のやり取りはしない。この様にノードを並べたものを**層**という。この層をミルフィーユ状に並べる。層を並べれば次に端の方から「第一層」、「第二層」、$\cdots$ と番号を振る。

次に層の間のシナプスをつなげていこう。まず、第一層の 1 つのノードに注目する。この時、このノードは第二層のノードと全部シナプスで繋げる。但し、シナプスは第一層のノードから第二層のノードへという向きに情報が伝わるもののみとする。同じようにほかの第一層のノードも第二層のノードとつなげてやる。同じように第二層と第三層をつなげ、どんどん順につなげていく。このようにして「層構造」を持ったニューラルネットワークを作ることができた。このようなものの中でも層の数が多いものを「深層学習」というのである（図 1.4)。

情報がどのようにニューラルネットを伝わっていくのかを見ていこう。まず、第一層を入力 $\boldsymbol{x}$ の状態にする。但し、第一層は複数のノードからなるのでベクトル表記で表される。次にシナプスが次のようなふるまいだったことを思い出そう。

$$y = \sigma(wx + b) \tag{1.4}$$

但し、$\sigma(x)$ は一般の非線形関数を表し、**活性化関数 (activate function)** という。第一層と第二層との間の関係を一つの式でまとめると次のようになる。

$$\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{1.5}$$

この時、$\boldsymbol{W}$ は行列、$\boldsymbol{y}, \boldsymbol{b}$ はベクトルである。$\boldsymbol{W}$ の (i, j) 成分は第一層の j 番目のノードと第二層の i 番目のノードの間のシナプスの重みであり、$\boldsymbol{b}$ の i 成分は第二層の i 番目の

ノードの発火の閾値である。情報の伝わり方を見たとき重要なのは第一層から順番に伝わっていくことである。

情報が順番に伝わるということはどこでどの様に情報が変換されたのかを辿ることができる。なのである適切な出力をするように重みやバイアスを調節しようとするときにどこをどの様に調節すればよいのかが分かりやすい。具体的には誤差逆伝播法というアルゴリズムで行われるが説明はのちに行う。

以上の説明で「層構造」と「ニューラルネットワーク」の意味が分かったと思う。もう一度深層学習の定義を振り返る。深層学習とは「深い層構造」を持った「ニューラルネットワーク」を機構として持つ機械学習のことである。

1.5　学習

深層学習で用いるアーキテクチャがわかった。次にそれがどの様に「学習」するか説明する。

まず、深層学習において「学習」とは何を意味するのか。もう一度犬猫判別機でのケースを思い出そう。この時の学習とは次のような関数 $Q(\boldsymbol{s})$ を求めることである。

$$Q(\boldsymbol{s}) \simeq P(cat|\boldsymbol{s}) \tag{1.6}$$

但し、$\boldsymbol{s}$ は画像データを表すベクトルであり、$P(cat|\boldsymbol{s})$ は画像 $\boldsymbol{s}$ が猫の画像である確率を表す。

つまりニューラルネットワークの重み $\boldsymbol{W}$ やバイアス $\boldsymbol{b}$ などのパラメータを調節し、データ分布 $P(cat|\boldsymbol{s})$ に近いような関数 $Q(\boldsymbol{s})$ を求めることが学習の定義となる。但し、必ずしも出力が $Q(\boldsymbol{s})$ とはならないことに注意しよう。例えば、明日の気温が知りたいとき確率分布が出力されるよりは予想気温を出力してくれた方が便利である。ただ、確率分布が分かれば予想気温はそれの平均値となるので本質的には同じことで、学習の定義は変わらない。

1.6　損失関数

次に問題となるのが「$Q(\boldsymbol{s})$ が $P(cat|\boldsymbol{s})$ に近づく」ということを定量的にどう扱うかである。ここで次のような確率分布を定義する。

$$Q(cat, \boldsymbol{s}) = Q(\boldsymbol{s})P(\boldsymbol{s}) \tag{1.7}$$

$$Q(dog, \boldsymbol{s}) = (1 - Q(\boldsymbol{s}))P(\boldsymbol{s}) \tag{1.8}$$

新しく定義した $Q(animal, \boldsymbol{s})$ は深層学習が $P(animal, \boldsymbol{s})$ を模倣しようとして作った確率分布としてみることができる。実際に適当に選んだ画像データを深層学習に判別させ、その結果に従って画像にラベル付けして作ったデータはデータ分布 $Q(animal, \boldsymbol{s})$ となる。

この時学習を $Q(animal, \boldsymbol{s})$ を $P(animal, \boldsymbol{s})$ に近づけることと言い換えることができる。そしてこの様な確率分布の間に「距離」のようなものを導入する。次のように定義される量を確率分布 $P(x)$ と $Q(x)$ の**カルバック・ライブラー距離**という。

$$D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{1.9}$$

なぜこれが「距離」とみなされるのか直感的理解の方法として様々な説明の仕方があるがここでは符号理論的な説明をする。

効率的に情報を伝達しようとするとき、ある事象が起こったことを伝えるのに用いる信号の数はその事象が起きる確率が大きいほど短くなることが一般的に知られている。

例えば、相手に待ち合わせ場所を伝えようとして電話を掛けたが相手は出なかった。そこであなたは留守電で待ち合わせ場所についてのメッセージを残しておこうと思った。しかし、留守電は時間が限られておりあまり長いメッセージは残すことができない。あなたはどんなメッセージを残すか。符号理論的には確率的に珍しいことを伝え、当たり前なことは削るとよい。例えば、京大生同士なら「京都大学の時計塔前集合」と言わなくても「時計塔前集合」と言えばいい。なぜなら京大生が他のロンドンの時計塔や札幌の時計塔に行く確率は低く、京大の時計塔に行く確率が高いからである。(図 1.5)

そして符号理論の定理により確率 P_i で起きる事象を伝えるのに用いるべき信号の長さは $-\log P_i$ に比例することがわかっている。信号の長さ $-\log P_i$ のことを**理想符号長**といい、$\sum_i -P_i \log P_i$ のことを理想的な**平均符号長**という。

この理論を応用して効率的に情報を伝達したい人がいた時を考えよう。用いるべき符号長は $-\log P_i$ で決まるが実際には真の確率分布 P_i を知ることはできない。そこで何らかの方法で確率分布 P_i に近い確率分布 Q_i を予想し、それを用いて符号化すると符号長は $-\log Q_i$ となる。この状況でカルバック・ライブラー距離の定義を見てみる。

$$D(P||Q) = \left[\sum_i -P_i \log Q_i\right] - \left[\sum_i -P_i \log P_i\right] \tag{1.10}$$

これは Q_i を用いたために理想的な符号長よりどれだけ無駄が増えたのかを表す量である。この量は必ず非負であり、また 0 となるのは $Q_i = P_i$ の時だけであることが証明できる。

このように符号理論の文脈から見ると $D(P||Q)$ という量は確率分布 Q_i をどれだけ P_i に近づけることができたのかの指標として有用であることがわかる。犬猫判別機の場合だ

図 1.5 待ち合わせをする京大生の図

と次のようになる。但し、以下 $a = animal$ とする。

$$\begin{aligned} D(P||Q) &= \sum_{a,\boldsymbol{s}} P(a,\boldsymbol{s}) \log \frac{P(a,\boldsymbol{s})}{Q(a,\boldsymbol{s})} \\ &= \sum_{a,\boldsymbol{s}} P(a,\boldsymbol{s}) \log \frac{P(a|\boldsymbol{s})P(\boldsymbol{s})}{Q(a|\boldsymbol{s})P(\boldsymbol{s})} \\ &= \sum_{a,\boldsymbol{s}} [P(a,\boldsymbol{s}) \log P(a|\boldsymbol{s}) - P(a,\boldsymbol{s}) \log Q(a|\boldsymbol{s})] \end{aligned}$$

パラメータを変化させて $Q(a,\boldsymbol{s})$ を調整したいので、パラメータに依存しない第一項については無視する。

$$\begin{aligned} D'(P||Q) &= \sum_{a,\boldsymbol{s}} -P(a,\boldsymbol{s}) \log Q(a|\boldsymbol{s}) \\ &\simeq \frac{1}{N_{data}} \sum_{data:i} -\log Q(a_i|\boldsymbol{s_i}) \\ &= \frac{1}{N_{data}} \sum_{data:i} -t_i \log Q(\boldsymbol{s_i}) - (1-t_i) \log (1 - Q(\boldsymbol{s_i})) \end{aligned} \tag{1.11}$$

但し、

$$t_i = \begin{cases} 1 & (a_i = cat) \\ 0 & (a_i = dog) \end{cases} \tag{1.12}$$

また二つ目の $\simeq$ の部分は N_{data} 個のデータによって $P(a,\boldsymbol{s})$ の近似をしている。式 (1.11) より学習とは与えられたデータに対して次の量をできるだけ小さくするようなパラメータを見つけることである。

$$L = -t_i \log Q(\boldsymbol{s_i}) - (1-t_i) \log (1 - Q(\boldsymbol{s_i})) \tag{1.13}$$

L のことを**損失関数**という。

1.7 勾配法

深層学習における学習とは損失関数の値が小さくなるようにパラメータを調整することであると定義した。根気強ければあらゆるパラメータの値を試してみてその中でも最小となるものを見つければよい。しかし、深層学習で調整しなければならないパラメータの数は非常に多く現実的ではない。そこで用いるのが勾配法というアルゴリズムである。これは損失関数のパラメータ微分の値を用いることで徐々にパラメータを改善していくことができるアルゴリズムである。

例えば、あなたは暗い夜に山登りをしているとしよう。手元に灯りや地図はない。視界で確認できるのはせいぜい半径 3 メートルの範囲だけだとする。この状況で山頂を目指す

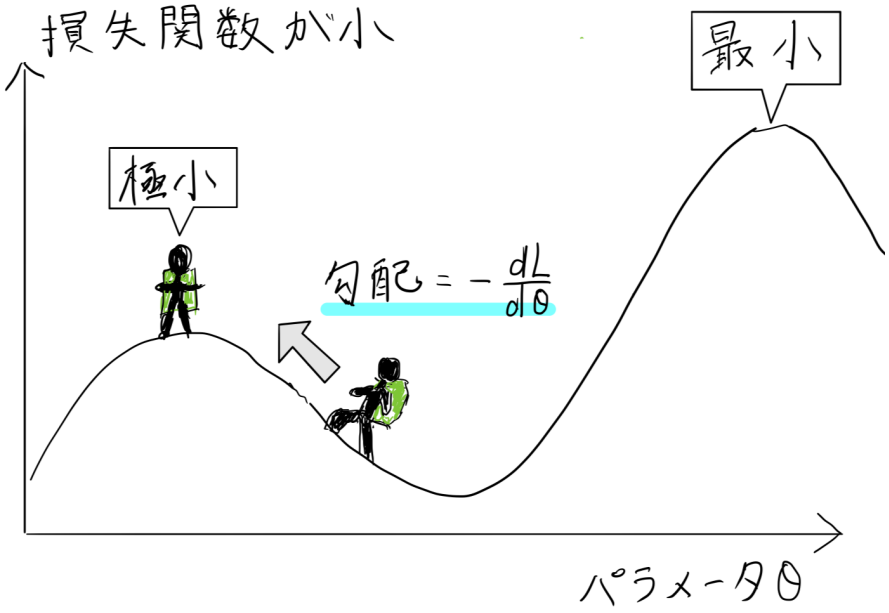


図 1.6 山登りと損失関数最小化とのアナロジー. 損失関数の値の低さと標高の高さとが対応している

としたらどのようにすればよいか。一つ考えられるのは坂が上がっていく方向に進む方法である。目指すのが一番高い場所なんだから、今確認できる範囲で標高が上がっていく方向に進めばいいというシンプルな考え方である。(図 1.6)

今の山登りのたとえのように勾配が一番きつい方向に進んでいって、損失関数の値が最小になるようなパラメータを見つけようとするアルゴリズムを**勾配法**という。この勾配法によって t 回更新されたパラメータを θ_t とすると、次のようになる。

$$\theta_{t+1} = \theta_t - \eta \frac{\partial L}{\partial \theta}(\theta_t) \tag{1.14}$$

ここの η を**学習率**といい予め設定しておくアルゴリズムのパラメータである。ちなみに学習率や層の深さのように勾配法などで更新されることがなく人間が設定したりするパラメータのことを**ハイパーパラメータ**という。先ほどの登山の例で言えば学習率は一歩の大きさのようなものである。

しかしここで注意しなければならないのは、勾配法で損失関数が最小になるパラメータが必ずしも見つかるとは限らないことである。日本で最も標高が高い場所は富士山山頂であるが、京都大学校門からスタートした人が勾配法で進んでいってもせいぜい吉田山が関の山である。それを改善しようとして作られた手法が多数あるが割愛する。

1.8 誤差逆伝播法

勾配法によってパラメータ調整ができるようになったが更新するためには損失関数の勾配を計算する必要がある。勾配とは数学の言葉でいえば微分の値なのでそれを求めればよい。しかし事態はそう簡単ではない。というのも深層学習で用いるパラメータの数がとても多いため勾配を求めるのも一苦労となる。しかし、層構造であることを利用すると勾配を効率よく求めることができる。これが**誤差逆伝播法**である。

順伝播の式 (1.5) を使えば入力 $\boldsymbol{x}$ に対して出力 $\boldsymbol{y}$ は次のように求められる。

$$\begin{aligned}\boldsymbol{h}_1 &= \sigma_1(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1)\\ \boldsymbol{h}_2 &= \sigma_2(\boldsymbol{W}_2\boldsymbol{h}_1 + \boldsymbol{b}_2)\\ \boldsymbol{h}_3 &= \sigma_3(\boldsymbol{W}_3\boldsymbol{h}_2 + \boldsymbol{b}_3)\\ \boldsymbol{y} &= \sigma_4(\boldsymbol{W}_4\boldsymbol{h}_3 + \boldsymbol{b}_4)\end{aligned}$$

そして学習の際の正解データを $\boldsymbol{d}$ とすると損失関数は $\boldsymbol{d}$ と出力 $\boldsymbol{y}$ との差を計るので

$$L = L(\boldsymbol{d}, \boldsymbol{y}) \tag{1.15}$$

となる。このとき、注目すべきなのは損失関数の値が入れ子のように $\boldsymbol{W}_i$ や $\boldsymbol{b}_i$ の値に依存していることである。つまり $L(\boldsymbol{d}, \boldsymbol{y})$ は直接は $\boldsymbol{y}$ に依存しているが次のように展開することもできる。

$$\begin{aligned}L(\boldsymbol{d}, \boldsymbol{y}) &= L(\boldsymbol{d}, \boldsymbol{y}(\boldsymbol{h}_3, \boldsymbol{W}_4, \boldsymbol{b}_4))\\ &= L(\boldsymbol{d}, \boldsymbol{y}(\boldsymbol{h}_3(\boldsymbol{h}_2, \boldsymbol{W}_3, \boldsymbol{b}_3), \boldsymbol{W}_4, \boldsymbol{b}_4))\\ &= L(\boldsymbol{d}, \boldsymbol{y}(\boldsymbol{h}_3(\boldsymbol{h}_2(\boldsymbol{h}_1, \boldsymbol{W}_2, \boldsymbol{b}_2), \boldsymbol{W}_3, \boldsymbol{b}_3), \boldsymbol{W}_4, \boldsymbol{b}_4))\\ &= L(\boldsymbol{d}, \boldsymbol{y}(\boldsymbol{h}_3(\boldsymbol{h}_2(\boldsymbol{h}_1(\boldsymbol{x}, \boldsymbol{W}_1, \boldsymbol{b}_1), \boldsymbol{W}_2, \boldsymbol{b}_2), \boldsymbol{W}_3, \boldsymbol{b}_3), \boldsymbol{W}_4, \boldsymbol{b}_4))\end{aligned}$$

またグラフで表すと図（1.7）のようになる。情報が順番に層の間で伝わっていく様子がこのような関数の入れ子のような形で表れているのである。この入れ子のような関係を考慮すると微分の連鎖律の性質により比較的単純な構造に分解できることがわかる。例えば、

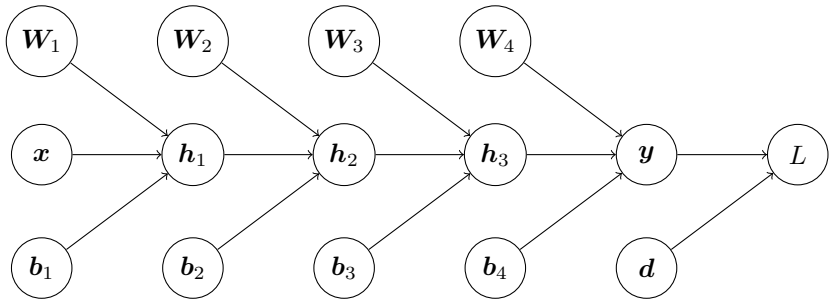


図 1.7 各変数及び関数の依存関係

パラメータである $\boldsymbol{b}_2$ の a 成分 b_2^a に注目すると

$$\begin{aligned}\frac{\partial L}{\partial b_2^a} &= \sum_i \frac{\partial L}{\partial y^i}\frac{\partial y^i}{\partial b_2^a} \\ &= \sum_i \frac{\partial L}{\partial y^i}\left(\sum_j \frac{\partial y^i}{\partial h_3^j}\frac{\partial h_3^j}{\partial b_2^a}\right) \\ &= \sum_i \frac{\partial L}{\partial y^i}\left\{\sum_j \frac{\partial y^i}{\partial h_3^j}\left(\sum_k \frac{\partial h_3^j}{\partial h_2^k}\frac{\partial h_2^k}{\partial b_2^a}\right)\right\}\end{aligned} \tag{1.16}$$

この時注目すべきなのは微分の値が自分が今いる場所よりも（順伝播的に見て）先の部分の微分の値から求められるということだ。これは微分と図（1.7）の矢印に対応させるとわかりやすい。例えば、$\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{b}_2}$ という微分と図（1.7）の $\boldsymbol{b}_2$ から $\boldsymbol{h}_2$ に向かう矢印と対応させる。この時、$\frac{\partial L}{\partial \boldsymbol{b}_2}$ の値は $\boldsymbol{b}_2$ から L までを結ぶ道が通る矢印が対応する微分の値の和積で表すことができるのである。また、道は好きなところで区切ることもできる。$\boldsymbol{b}_2$ から L までの道を $\boldsymbol{b}_2$ から $\boldsymbol{h}_2$ までの道と $\boldsymbol{h}_2$ から $\boldsymbol{b}_2$ までの道に区切ったときそれぞれが対応する微分の和積で元々の道が対応していた微分の値を表すことができるである。

以上の事実から全てのパラメータについての勾配を知ろうとしたとき、効率的な求め方が図（1.7）のグラフの形からおのずと見えてくる。損失関数から近いパラメータの勾配から調べていけばよいのである。はじめはパラメータから損失関数までの距離が近く勾配計算が楽である。次々と調べていくうちに段々距離が長くなっていくが、道の大部分がもうすでに調べてある道であり、その結果を用いれば比較的容易に計算することができる。このようにして勾配を求めていく方法を**誤差逆伝播法**という。この名前は誤差についての情報が後ろから伝播してきていることを表す。

また誤差逆伝播法を知ることで、層構造を導入した意味が分かってもらえたかと思う。例えば、蜘蛛の巣のようなループを持ったニューラルネットワークを考えるとパラメータ全体を更新するということができないことがわかるだろう。というのも、そのような構造の場合、順伝播や逆伝播のような情報の伝わり方の向きのようなものが定義できないので「誤差逆伝播法」は用いることができない。層構造を導入することによって全パラメータを「同時に」更新することができるのである。

1.9 まとめ

以上が深層学習の基本的な構造である。全体をもう一度振り返る。まず深層学習とは「深い層構造」をもった「ニューラルネットワーク」をもった機械学習の一種である。ニューラルネットワークは脳の神経細胞を模したネットワークであり、数理的には線形変換と非線形変換の繰り返しにより情報の伝達が再現されている。深層学習ではそれが層構造をなしている。層構造とはノードが層状に並び、層と層との間はノード同士で繋がり合っている。

こうした深層学習はパラメータを調整することで学習をする。調整の基準となるのが損失関数であり、これが小さくなるようにすればよい。調整方法としては勾配法を用いる。これは自分が今いる場所の坂の傾きに逆らって進むことで暗闇の中、山頂を目指すようなアルゴリズムである。しかし、いざ勾配法でパラメータ調整をしようとしても、肝心の勾配計算が大変である。これを解決するのが誤差逆伝播法である。これは入力から出力へと順番に情報が伝わっているのだから、出力のずれのパラメータ補正は逆向きに行えばいいというものである。具体的には微分の連鎖律などによって正当化される。

最後に現在なぜ深層学習が多くの場面で活躍しているのか、深層学習の構造という観点から説明したいと思う。

一つ目の理由としてはシンプルな構造ゆえのモデルの可変性の高さが挙げられる。深層学習は図（1.4）のような形をしていることから層の数や幅を増やしたり、入力層と出力層だけを別のものと入れ替えたりすることで様々なモデルを作ることができる。これによって様々なデータの形に対応できる。例えば、時系列データと呼ばれる会話や音楽のようなデータと画像データはデータの形として全く違うものなので普通なら全く別の機械学習を用いて処理することが多い。だが深層学習はモデルの可変性の高さゆえに対応することができるのである。

二つ目の理由は表現能力の高さである。深層学習ではあまりよくわかってないデータを処理したいことが多いので、あらゆるデータ分布の形に対応できるような能力を持っていることが望ましい。実際、深層学習ではその要請を満たしていて深層化することによって表現能力を高めているのである。直感的にはパラメータの数が増えたので表現能力が上

がったということである。但し、線形変換だけの時は深層化しても意味が無い。簡単に言えば 2 かけた後に 3 かけるのは 6 を一回かけるのと同じ操作であるのでその分パラメータによる表現能力が死ぬからである。そこで重要になるのが非線形の活性化関数である。活性化関数は往々にして図（1.3）のように階段のような形をしているが、これを×２と×３との操作の間に挟めば 2 かけた後の数値が閾値より大きいか、小さいかで結果が全く異なってくる。このように活性化関数のおかげで深層化すると高い表現能力を獲得することができるのである。

深層学習が画像処理、画像生成、言語処理、創薬など多方面で活躍しているわけを大まかにわかっていただけたかと思う。だからといって実際に使うとなると課題に合わせた工夫が必要であり、そしてそれが深層学習の面白みの一つである。例えば、畳み込みニューラルネットワーク (Convolutional Neural Network,CNN) と呼ばれる画像処理用のモデルでは実際の人間の視覚に関する神経科学の知見が生かされていている。他にも、敵対的生成ネットワーク (Generative Adversarial Network,GAN) と呼ばれる画像生成のモデルでは二つの深層学習が一方が偽物の画像を作り、他方がそれが偽物の画像かどうかを判別するというゲームを繰り返せば、最終的には精度のいい画像が作ることができるようになるのではないかというアイデアを基にして作られている。

深層学習の理論的な解析はとても複雑でまだよくわかっていないことが多い。実際に動かすことができるようになったのも最近である。ただ、深層学習が高い能力を持っていることは現在の活躍ぶりを見れば明らかである。何かちょっとしたアイデアで深層学習のいまだ知られざる一面を明らかにすることができるかもしれない。深層学習の可能性は広く、その世界はずっと深い。

第 2 章

機械学習のための偏微分入門

2.1　はじめに

機械学習やディープラーニングの書籍でしばしば登場するベクトルや行列での偏微分について, 基礎をまとめた. 学習や研究に役立ててもらいたい.

2.2　ベクトルと行列に関する記号の定義

定義 2.2.1 ベクトル $\boldsymbol{x}$ が $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ と表されるとき,

$$\|\boldsymbol{x}\| \stackrel{\mathrm{def}}{=} \sqrt{|x_1|^2 + \cdots + |x_n|^2}$$

を $\boldsymbol{x}$ のユークリッドノルムという.

定義 2.2.2 行列 A が $A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}$ と表されるとき,

$$\|A\|_{\mathrm{F}} \stackrel{\mathrm{def}}{=} \sqrt{\sum_{i,j} |a_{ij}|^2}$$

を A のフロベニウスノルムという.

定義 2.2.3 ベクトル $\boldsymbol{x}$, $\boldsymbol{y}$ が $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ と表されるとき,

$$\boldsymbol{x} \odot \boldsymbol{y} \stackrel{\mathrm{def}}{=} (x_1 y_1, x_2 y_2, \ldots, x_n y_n)^{\mathrm{T}}$$

を $\boldsymbol{x}$ と $\boldsymbol{y}$ のアダマール積という.

定義 2.2.4 ベクトル $\boldsymbol{x}$ が $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ と表されるとき, $\mathrm{diag}(\boldsymbol{x})$ は

$$\mathrm{diag}(\boldsymbol{x}) \stackrel{\mathrm{def}}{=} \begin{pmatrix} x_1 & 0 & \ldots & 0 \\ 0 & x_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & x_n \end{pmatrix}$$

という対角行列を表す[*1].

定義 2.2.5

$$\delta_{ij} \stackrel{\mathrm{def}}{=} \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

を**クロネッカーのデルタ**という.

定義 2.2.6 行列の添字記法を定義する.

- $(\text{行列})_{ij}$ は行列の i 行 j 列成分を表す.
- (x_{ij}) は i 行 j 列の成分が x_{ij} であるような行列を表す[*2].

例 1

$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ のとき, 行列 A の 1 行 2 列の成分は 2 なので, $A_{12} = 2$.

例 2

$x_{11} = 1, x_{12} = 2, x_{21} = 3, x_{22} = 4$ のとき $(x_{ij}) = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $(x_{ji}) = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$.

例 3

行列 A と行列 B の積 AB は次のように書くことができる.

$$AB = (A_{ij})(B_{ij}) = \left(\sum_k A_{ik} B_{kj} \right)$$

[*1] ベクトルを対角行列化する演算子 diag を用いれば、アダマール積は次のように表すことができる.

$$\begin{aligned} \boldsymbol{x} \odot \boldsymbol{y} &= \mathrm{diag}(\boldsymbol{x})\boldsymbol{y} \\ &= \mathrm{diag}(\boldsymbol{y})\boldsymbol{x} \end{aligned}$$

[*2] i が行を指し j が列を指すと約束する.

2.3 定義

定義 2.3.1（一変数関数の微分） ある区間において, 変数 x の関数 $y = f(x)$ が与えられているとする. このとき, 区間内の固定された点 x に対し,

$$\lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x) - a\Delta x}{\Delta x} = 0 \tag{2.1}$$

を満たす定数 a が存在するとき, $f(x)$ は**点 x において微分可能**であるという. また, この定数 a を点 x における $f(x)$ の**微分係数**と呼ぶ. ある区間の各点 x において $f(x)$ が微分可能ならば, $f(x)$ はその**区間において微分可能**であるという. その場合, 微分係数 a は x の関数である. その関数を $f(x)$ の**導関数**といい, $\frac{dy}{dx}, f'(x), y', \dot{y}, D_x y, Df(x)$ などと表す。

偏微分

二つ以上の引数をとる関数において, ただ一つの変数のみを変動させて, その変数に関して微分する[*3]ことを**偏微分**という.

定義 2.3.2（偏微分） ある領域において, 変数 $x_1, x_2, \ldots, x_n$ の関数 $y = f(x_1, x_2, \ldots, x_n)$ が与えられているとする. このとき, 領域内の固定された点 $(x_1, x_2, \ldots, x_n)$ に対し, 極限

$$\lim_{\Delta x \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + \Delta x, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_n)}{\Delta x} \tag{2.2}$$

が存在するとき, $f(x_1, \ldots, x_n)$ は**点 $(x_1, \ldots, x_n)$ において x_i に関して偏微分可能**であるという. また, この極限を**点 $(x_1, \ldots, x_n)$ における x_i に関する偏微分係数**という. ある領域の各点 $(x_1, \ldots, x_n)$ において x_i に関して偏微分可能ならば, $f(x_1, \ldots, x_n)$ はその**領域において x_i に関して偏微分可能**であるという. その場合, 極限 (式 2.2) は $x_1, \ldots, x_n$ の関数である. その関数を $f(x_1, \ldots, x_n)$ の x_i に関する**偏導関数**といい, $\frac{\partial y}{\partial x}, f_{x_i}(x_1, \ldots, x_n), y_{x_i}, D_{x_i} f(x_1, \ldots, x_n)$ などと表す.

定義 2.3.3（スカラーのベクトル微分） ある領域において, 変数 $x_1, x_2, \ldots, x_n$ の関数 $y = f(x_1, x_2, \ldots, x_n)$ が与えられているとする. このとき, ベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ に対して,

$$\left(\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_n} \right) \tag{2.3}$$

[*3] 微分係数や導関数を求めることを**微分する**という. x_i に関する偏微分係数や偏導関数を求めることを x_i **に関して偏微分する**という.

を f の勾配といい, $\frac{\partial y}{\partial \boldsymbol{x}}, \nabla f, \mathrm{grad} f$ などと表す[*4].

定義 2.3.4（ベクトルのベクトル微分） ある領域において, 変数 $x_1, x_2, \ldots, x_n$ のベクトル値関数 $\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \boldsymbol{y} = \boldsymbol{f}(x_1, x_2, \ldots, x_n) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \end{pmatrix}$ が与えられているとする. このとき, ベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ に対して,

$$\begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_j} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y_i}{\partial x_1} & \cdots & \frac{\partial y_i}{\partial x_j} & \cdots & \frac{\partial y_i}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_j} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix} \tag{2.4}$$

を $\boldsymbol{f}$ のヤコビ行列といい, $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}, J_{\boldsymbol{f}}, D_{\boldsymbol{x}} \boldsymbol{f}$ などと表す[*5].

定義 2.3.5（スカラーの行列微分） ある領域において, 行列 A を引数にとる関数 $y = f(A)$ が与えられているとする. このとき, 行列 A に対して,

$$\begin{pmatrix} \frac{\partial y}{\partial A_{11}} & \cdots & \frac{\partial y}{\partial A_{1j}} & \cdots & \frac{\partial y}{\partial A_{1n}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial A_{i1}} & \cdots & \frac{\partial y}{\partial A_{ij}} & \cdots & \frac{\partial y}{\partial A_{in}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial A_{m1}} & \cdots & \frac{\partial y}{\partial A_{mj}} & \cdots & \frac{\partial y}{\partial A_{mn}} \end{pmatrix} \tag{2.5}$$

を行列微分といい, $\frac{\partial \boldsymbol{y}}{\partial A}, \nabla_A f$ などと表す[*6][*7].

[*4] 引数のベクトルは列ベクトルであるのに対し, 勾配は行ベクトルとして定義していることに注意する.

[*5] この定義では i 行目に出力の y_i が現れ, j 列目に入力の x_j が現れるようになっている. シンプルな表し方である $\left(\frac{\partial y_i}{\partial x_j}\right)$ で覚えるとよい. ヤコビ行列と勾配は, ヤコビ行列が勾配の拡張となるように, かつ, $d\boldsymbol{y} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} d\boldsymbol{x}$ が成立するように定義されている. この定義は Numerator-layout notation であるが, 行と列を入れ換えて定義することもある [3].

[*6] 行列微分は一般的な数学用語ではないが, 行列微分の操作自体は機械学習の書籍にしばしば現れる. 例えば “Deep Learning” (MIT Press) の日本語版では, この操作のことを「行列の微分」という言葉で表している.『パターン認識と機械学習 上』の付録 C でも「行列の微分」の説明の中にこの操作が現れる. ただし『パターン認識と機械学習 上』ではこの操作に対する名称は与えられておらず, 行列に関する微分の一つとして紹介されている.

[*7] 以上のベクトルや行列に対する微分の定義は, $\frac{\partial y}{\partial \boldsymbol{x}} = \left(\frac{\partial y}{\partial x_j}\right), \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} = \left(\frac{\partial y_i}{\partial x_j}\right), \frac{\partial y}{\partial A} = \left(\frac{\partial y}{\partial A_{ij}}\right)$ と表すことができる.

定義 2.3.6（全微分） ある領域において, 変数 $x_1, x_2, \ldots, x_n$ のベクトル値関数 $\boldsymbol{f}(x_1, x_2, \ldots, x_n) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_m(x_1, x_2, \ldots, x_n) \end{pmatrix}$ が与えられているとする. このとき, 領域内の固定された点 $(x_1, x_2, \ldots, x_n)$ に対し,

$$\lim_{\|\Delta \boldsymbol{x}\| \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x} + \Delta \boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - A\Delta \boldsymbol{x}\|}{\|\Delta \boldsymbol{x}\|} = 0 \tag{2.6}$$

を満たす行列 A が存在するとき, $\boldsymbol{f}(\boldsymbol{x})$ は**点 $\boldsymbol{x}$ において全微分可能**であるという[*8][*9]. また, この行列 A を点 $\boldsymbol{x}$ における $\boldsymbol{f}(\boldsymbol{x})$ の**全微分係数**と呼ぶ. ある領域の各点 $\boldsymbol{x}$ において $\boldsymbol{f}(\boldsymbol{x})$ が全微分可能ならば, $\boldsymbol{f}(\boldsymbol{x})$ はその**領域において全微分可能**であるという.

[*8] (式 2.6) の分子は m 次元ベクトルのユークリッドノルムであり, 分母は n 次元ベクトルのユークリッドノルムである.

[*9] $\lim_{\|\Delta \boldsymbol{x}\| \to 0}$ はベクトル $\Delta \boldsymbol{x}$ のノルム (長さ) を小さくしていくときの極限を表す. 極限が存在するには, 様々な方向の $\Delta \boldsymbol{x}$ をノルムが 0 になるように近づけたときに lim の中身が同じ値に近づく必要がある.

2.4　公式

一変数関数の微分に関して次の等式が成り立つことが知られている. ここで a, b は定数で, f, g は x の関数, $\exp x$ は指数関数 e^x で, $\ln x$ は自然対数 $\log_{\mathrm{e}} x$ である. e はネイピア数 $\mathrm{e} = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$ を表す.

$$\frac{d(af + bg)}{dx} = a\frac{df}{dx} + b\frac{dg}{dx} \tag{2.7}$$

$$\frac{dfg}{dx} = \frac{df}{dx}g + f\frac{dg}{dx} \tag{2.8}$$

$$\left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2} \quad (g \neq 0) \tag{2.9}$$

$$(\exp x)' = \exp x \tag{2.10}$$

$$(\ln x)' = \frac{1}{x} \quad (x > 0) \tag{2.11}$$

2.5 問題

1 (微分係数の一意性) 次の問いに答えよ.

(1) 一変数関数の微分係数 a が存在する場合, 微分係数 a は $a = \lim_{\Delta x \to 0} \dfrac{f(x+\Delta x) - f(x)}{\Delta x}$ で与えられることを示せ[*10].

(2) ベクトル値関数の点 $\boldsymbol{x}$ における全微分係数 A が存在する場合, 全微分係数 A は $A = D_{\boldsymbol{x}}\boldsymbol{f}$ で与えられることを示せ[*11].

2 (連鎖律) 次の問いに答えよ.

(1) 関数 z は $z = z(y(x))$ で与えられており, x のとりえる区間は定められている. z は $y = y(x)$ のとりえる区間において微分可能で, y は x のとりえる区間において微分可能であるとする. このとき, z が x のとりえる区間において微分可能であり, 次式を満たすことを示せ.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

(2) 関数 z は $z = z(\boldsymbol{y}(x))$ で与えられており, x のとりえる区間は定められている. z は $\boldsymbol{y} = \boldsymbol{y}(x)$ のとりえる領域において全微分可能で, $\boldsymbol{y}$ は x のとりえる区間において微分可能であるとする. このとき, z が x のとりえる区間において微分可能であり, 次式を満たすことを示せ[*12].

$$\frac{dz}{dx} = \sum_k \frac{\partial z}{\partial y_k}\frac{dy_k}{dx}$$

(3) 関数 z は $z = z(\boldsymbol{y}(\boldsymbol{x}))$ で与えられており, $\boldsymbol{x}$ のとりえる領域は定められている. z は $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ のとりえる領域において全微分可能で, $\boldsymbol{y}$ は $\boldsymbol{x}$ のとりえる領域において全微分可能であるとする. このとき, z が $\boldsymbol{x}$ のとりえる領域において全微分可能であり, 次式を満たすことを示せ[*13].

$$\frac{\partial z}{\partial x_j} = \sum_k \frac{\partial z}{\partial y_k}\frac{\partial y_k}{\partial x_j}$$

[*10] このことから, 微分係数が存在する場合, 微分係数は一意的に定まることがいえる.

[*11] このことから, 全微分係数が存在する場合, 全微分係数は一意的に定まることがいえる.

[*12] 一変数関数の微分は偏微分の特別な形とみることができるので, (2) から (1) は直ちに得られる.

[*13] 一変数関数の微分は偏微分の特別な形とみることができるので, (3) から (1) , (2) は直ちに得られる.

3 (基本公式) 次の等式中の各演算が定義できるとき, 等式が成立することを示せ. ただし, $\boldsymbol{x}$ はベクトル, X は行列, f, g は実数値関数, $\boldsymbol{f}$, $\boldsymbol{g}$ はベクトル値関数, a, b は定数 (スカラー), A, B は定数行列とする.

(1)
$$\frac{\partial(af+bg)}{\partial \boldsymbol{x}} = a\frac{\partial f}{\partial \boldsymbol{x}} + b\frac{\partial g}{\partial \boldsymbol{x}}$$

(2)
$$\frac{\partial(Af+Bg)}{\partial \boldsymbol{x}} = A\frac{\partial f}{\partial \boldsymbol{x}} + B\frac{\partial g}{\partial \boldsymbol{x}}$$

(3)
$$\frac{\partial(a\boldsymbol{f}+b\boldsymbol{g})}{\partial \boldsymbol{x}} = a\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + b\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$$

(4)
$$\frac{\partial(A\boldsymbol{f}+B\boldsymbol{g})}{\partial \boldsymbol{x}} = A\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + B\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$$

(5)
$$\frac{\partial(af+bg)}{\partial X} = a\frac{\partial f}{\partial X} + b\frac{\partial g}{\partial X}$$

(6)
$$\frac{\partial fg}{\partial \boldsymbol{x}} = \frac{\partial f}{\partial \boldsymbol{x}}g + f\frac{\partial g}{\partial \boldsymbol{x}}$$

(7)
$$\frac{\partial \boldsymbol{f}^{\mathrm{T}}\boldsymbol{g}}{\partial \boldsymbol{x}} = \boldsymbol{g}^{\mathrm{T}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}^{\mathrm{T}}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}$$

4 (連鎖律をヤコビ行列で表示) 次の問いに答えよ.

(1) ベクトル値関数 $\boldsymbol{z}$ が $\boldsymbol{z} = \boldsymbol{z}(\boldsymbol{y}(\boldsymbol{x}))$ で与えられており, 等式中に現れる各演算が定義できるとき, 次の等式が成立することを示せ.

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$$

5 (ベクトルをベクトル微分) 次の等式が成り立つことを示せ. ただし, $\boldsymbol{x}$ はベクトル, I は単位行列, A は定数行列, f は一変数実数値関数, $\boldsymbol{a}$ は定数ベクトルである.

(1)
$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = I$$

(2)

$$\frac{\partial A\boldsymbol{x}}{\partial \boldsymbol{x}} = A$$

(3)

$$\frac{\partial \left(f(x_1), f(x_2), \ldots, f(x_n)\right)^{\mathrm{T}}}{\partial \boldsymbol{x}} = \begin{pmatrix} f'(x_1) & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & f'(x_i) & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & f'(x_n) \end{pmatrix}$$

(4)

$$\frac{\partial \boldsymbol{a} \odot \boldsymbol{x}}{\partial \boldsymbol{x}} = \mathrm{diag}(\boldsymbol{a})$$

6 (スカラーをベクトル微分) 次の等式が成り立つことを示せ. ただし, $\boldsymbol{x}$ はベクトル, A, C は定数行列, $\boldsymbol{a}$, $\boldsymbol{b}$ は定数ベクトルである.

(1)

$$\frac{\partial \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^{\mathrm{T}}$$

(2)

$$\frac{\partial \boldsymbol{x}^{\mathrm{T}}\boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}^{\mathrm{T}}$$

(3)

$$\frac{\partial \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{\partial \boldsymbol{x}} = 2\boldsymbol{x}^{\mathrm{T}}$$

(4)

$$\frac{\partial \boldsymbol{x}^{\mathrm{T}}A\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^{\mathrm{T}}(A + A^{\mathrm{T}})$$

(5)

$$\frac{\partial (\boldsymbol{x} + \boldsymbol{a})^{\mathrm{T}}(\boldsymbol{x} + \boldsymbol{a})}{\partial \boldsymbol{x}} = 2(\boldsymbol{x} + \boldsymbol{a})^{\mathrm{T}}$$

(6)

$$\frac{\partial (A\boldsymbol{x} + \boldsymbol{b})^{\mathrm{T}}(A\boldsymbol{x} + \boldsymbol{b})}{\partial \boldsymbol{x}} = 2(A\boldsymbol{x} + \boldsymbol{b})^{\mathrm{T}}A$$

(7)

$$\frac{\partial (A\boldsymbol{x} + \boldsymbol{b})^{\mathrm{T}}C(A\boldsymbol{x} + \boldsymbol{b})}{\partial \boldsymbol{x}} = (A\boldsymbol{x} + \boldsymbol{b})^{\mathrm{T}}(C + C^{\mathrm{T}})A$$

7 (スカラーを行列微分) 次の等式が成り立つことを示せ. ただし, X は行列, I は対角成分が全て 1 で他の成分が全て 0 の行列, A, B は定数行列, f は行列を引数にとる実数値関数である.

(1)
$$\frac{\partial \operatorname{tr}(X)}{\partial X} = I$$

(2)
$$\frac{\partial \operatorname{tr}(AX)}{\partial X} = A^{\mathrm{T}}$$

(3)
$$\frac{\partial \operatorname{tr}(XA)}{\partial X} = A^{\mathrm{T}}$$

(4)
$$\frac{\partial \operatorname{tr}(X^{\mathrm{T}}AX)}{\partial X} = (A + A^{\mathrm{T}})X$$

(5)
$$\frac{\partial f(X)}{\partial X^{\mathrm{T}}} = \left(\frac{\partial f(X)}{\partial X}\right)^{\mathrm{T}}$$

(6)
$$\frac{\partial \det(X)}{\partial X} = \det(X)(X^{-1})^{\mathrm{T}}$$

(7)
$$\frac{\partial \ln \det(X)}{\partial X} = (X^{-1})^{\mathrm{T}}$$

(8)
$$\frac{\partial \|AX + B\|_{\mathrm{F}}^2}{\partial X} = 2A^{\mathrm{T}}(AX + B)$$

(9)
$$\frac{\partial \|XA + B\|_{\mathrm{F}}^2}{\partial X} = 2(XA + B)A^{\mathrm{T}}$$

8 (単層ニューラルネットワークの勾配降下法を用いた学習)
$\boldsymbol{y} = W\boldsymbol{x} + \boldsymbol{b}$, $L = L(\boldsymbol{y})$ のとき, 以下の問いに答えよ. ただし, $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{b}$ はベクトル, W は行列である.

(1) $\dfrac{\partial L}{\partial W}$ を $\dfrac{\partial L}{\partial \boldsymbol{y}}$ と $\boldsymbol{x}$ を用いて表せ.

(2) $\dfrac{\partial L}{\partial \boldsymbol{b}}$ を $\dfrac{\partial L}{\partial \boldsymbol{y}}$ と $\boldsymbol{x}$ を用いて表せ.

9 (アフィンレイヤの誤差逆伝播)
$\boldsymbol{p} = W\boldsymbol{y} + \boldsymbol{b}$, $\boldsymbol{y}' = \boldsymbol{f}(\boldsymbol{p})$, $\dfrac{\partial L}{\partial \boldsymbol{y}'} = \boldsymbol{e}$ のとき, $\dfrac{\partial L}{\partial \boldsymbol{y}}$ を $\boldsymbol{e}$ と $\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}$ を用いて表せ. ただし, $\boldsymbol{y}$, $\boldsymbol{y}'$, $\boldsymbol{b}$ はベクトル, W は行列, $\boldsymbol{f}$ はベクトル値関数である.

10 (ソフトマックス関数+クロスエントロピー誤差の誤差逆伝播)

$$\begin{cases} y_k = \dfrac{\exp a_k}{\displaystyle\sum_l \exp a_l} \\ L = -\displaystyle\sum_k t_k \ln y_k \\ \displaystyle\sum_k t_k = 1 \end{cases}$$

のとき, $\dfrac{\partial L}{\partial a_j}$ を簡単な式で表せ.

11 (リカレントニューラルネットワークの誤差逆伝播)

$$\begin{cases} h_i(t-1) = f(p_i(t-1)) \\ \boldsymbol{p}(t) = U\boldsymbol{x}(t) + W\boldsymbol{h}(t-1) + \boldsymbol{b} \\ \boldsymbol{e}(t) = \dfrac{\partial L}{\partial \boldsymbol{p}(t)} \end{cases}$$

のとき, $\dfrac{\partial L}{\partial \boldsymbol{p}(t-1)}$ を求めよ.

12 (京大情報学研究科 修士課程 知能情報学専攻 入学者選抜試験 2018 年実施 T-3)

予測問題を考える. d 次元の入力ベクトルを $\boldsymbol{x} = [x^{(1)}, x^{(2)}, \ldots, x^{(d)}]^{\mathrm{T}} \in \mathbb{R}^d$ (T は転置), それに対応する出力を $y \in \mathbb{R}$ とする. 入力と出力が対になった学習データを $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ とし, それぞれの $(\boldsymbol{x}_i, y_i)$ $(i = 1, 2, \ldots, n)$ は独立に同一分布に従う. なお, データ行列 $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ のランクを d とする.

線形モデル

$$f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$$

を考え, $\boldsymbol{w} = [w^{(1)}, w^{(2)}, \ldots, w^{(d)}]^{\mathrm{T}} \in \mathbb{R}^d$ をモデルパラメータとする.

設問 1

以下の式を満たす $\hat{\boldsymbol{w}}$ を学習データを用いて導け.

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2$$

設問 2

出力 y_i と入力 $\boldsymbol{x}_i$ に次の関係があるとする.

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{w}) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2)$$

ここで $N(\mu, \sigma^2)$ は平均 μ, 分散 σ^2 の 1 次元の正規分布である. $\boldsymbol{w}$ の尤度関数を定義した上で, その最尤推定量を求めよ.

設問 3

以下の式を満たす $\hat{\boldsymbol{w}}$ を学習データを用いて導け.

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} r_i (y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|^2 \right)$$

なお, $r_i > 0$ $(i = 1, 2, \ldots, n)$ および $\lambda > 0$ は定数とし, $\|\cdot\|$ はユークリッドノルムとする.

設問 4

予測モデルの出力を $y \in \{0, 1\}$ とし, $y = 1$ の事後確率の確率モデルを

$$p(y = 1 | \boldsymbol{x}; \boldsymbol{w}) = \frac{1}{1 + \exp(-f(\boldsymbol{x}; \boldsymbol{w}))}$$

とする. $y = 0$ の事後確率の確率モデルを $\boldsymbol{x}$ と $\boldsymbol{w}$ を用いて表せ.

設問 5

学習データの出力を $y_i \in \{0, 1\}$ $(i = 1, 2, \ldots, n)$ とする. 設問 4 の確率モデルにおいて, 対数尤度関数 $L(\boldsymbol{w})$ の $\boldsymbol{w}$ に関する勾配が次で表せることを示せ.

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = \sum_{i=1}^{n} (y_i - p(y = 1 | \boldsymbol{x}_i; \boldsymbol{w})) \boldsymbol{x}_i$$

2.6 解答

1 (微分係数の一意性)

(1)

微分係数の定義式 (式 2.1) を変形していけばよい.

$$\begin{aligned} & \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x) - a\Delta x}{\Delta x} = 0 \\ \iff \quad & \lim_{\Delta x \to 0} \left(\frac{f(x+\Delta x) - f(x)}{\Delta x} - a \right) = 0 \\ \iff \quad & a = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \end{aligned}$$

であるから, 微分係数 a は, $a = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$ で与えられる[*14].

(2)

全微分係数の定義式 (式 2.6) を変形していけばよい.

全微分係数 A が存在するとする.

$$\begin{aligned} & \lim_{\|\Delta \boldsymbol{x}\| \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x}+\Delta \boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - A\Delta \boldsymbol{x}\|}{\|\Delta \boldsymbol{x}\|} = 0 \\ \iff \quad & \lim_{\|\Delta \boldsymbol{x}\| \to 0} \frac{\|\boldsymbol{f}(\boldsymbol{x}+\Delta \boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{x}) - A\Delta \boldsymbol{x}\|^2}{\|\Delta \boldsymbol{x}\|^2} = 0 \\ \iff \quad & \lim_{\|\Delta \boldsymbol{x}\| \to 0} \frac{\sum_k |f_k(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_k(\boldsymbol{x}) - (A_{kj})\Delta \boldsymbol{x}|^2}{\sum_k |\Delta \boldsymbol{x}|^2} = 0. \end{aligned} \tag{2.12}$$

(式 2.12) の $k = i$ となる項だけを取り出すと,

$$(\text{式 } 2.12) \Longrightarrow \lim_{\|\Delta \boldsymbol{x}\| \to 0} \frac{|f_i(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_i(\boldsymbol{x}) - \sum_l A_{il}\Delta x_l|^2}{\sum_k |\Delta \boldsymbol{x}|^2} = 0. \tag{2.13}$$

[*14] この同値変形より, この極限の存在と微分係数の存在は同値であることがわかる.

さらに, $\Delta \boldsymbol{x}$ の j 行目以外の要素はすべて 0 にし, j 行目の要素だけを動かして $\|\Delta \boldsymbol{x}\| \to 0$ となるように近づけると,

$$
\begin{aligned}
(\text{式 } 2.13) &\Longrightarrow \lim_{|\Delta x_j| \to 0} \frac{|f_i(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_i(\boldsymbol{x}) - A_{ij}\Delta x_j|^2}{|\Delta x_j|^2} = 0 \\
&\Longleftrightarrow \lim_{|\Delta x_j| \to 0} \frac{|f_i(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_i(\boldsymbol{x}) - A_{ij}\Delta x_j|}{|\Delta x_j|} = 0 \\
&\Longleftrightarrow \lim_{\Delta x_j \to 0} \frac{f_i(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_i(\boldsymbol{x}) - A_{ij}\Delta x_j}{\Delta x_j} = 0 \\
&\Longleftrightarrow A_{ij} = \lim_{\Delta x_j \to 0} \frac{f_i(\boldsymbol{x}+\Delta \boldsymbol{x}) - f_i(\boldsymbol{x})}{\Delta x_j}.
\end{aligned}
$$

この式の右辺は $\dfrac{\partial f_i}{\partial x_j}$ に等しい。

したがって、全微分係数 A は $A = (A_{ij}) = \left(\dfrac{\partial f_i}{\partial x_j}\right) = D_{\boldsymbol{x}}\boldsymbol{f}$ で与えられる[*15].

2 (連鎖律)

(1) 一変数関数の連鎖律

一変数関数の微分の定義式 (式 2.1) を利用すればよい.

はじめに, x を Δx だけ動かしたときの y の変位を Δy, z の変位を Δz とする. すなわち,

$$
\begin{cases}
\Delta y = y(x+\Delta x) - y(x) & (2.14) \\
\Delta z = z(y+\Delta y) - z(y) & (2.15)
\end{cases}
$$

とする. さて, y は x で微分可能で, z は y で微分可能なので,

$$
\begin{cases}
\displaystyle\lim_{\Delta x \to 0} \frac{y(x+\Delta x) - y(x) - \dfrac{dy}{dx}\Delta x}{\Delta x} = 0 & (2.16) \\
\displaystyle\lim_{\Delta y \to 0} \frac{z(y+\Delta y) - z(y) - \dfrac{dz}{dy}\Delta y}{\Delta y} = 0 & (2.17)
\end{cases}
$$

が成り立つ. ここで, $\varepsilon(x, \Delta x)$, $\varepsilon'(x, \Delta x)$ を次を満たすように定める.

$$
\begin{cases}
y(x+\Delta x) - y(x) - \dfrac{dy}{dx}\Delta x + \varepsilon(x, \Delta x)\Delta x = 0 & (2.18) \\
z(y+\Delta y) - z(y) - \dfrac{dz}{dy}\Delta y + \varepsilon'(x, \Delta x)\Delta y = 0. & (2.19)
\end{cases}
$$

[*15] 全微分係数 A が存在するとき A はヤコビ行列となるが, ヤコビ行列が存在するからといって全微分係数が存在するわけではないことに注意する. 実際, 点 $\boldsymbol{x}$ に対して全微分係数が存在することの必要十分条件は, ヤコビ行列が存在すること, かつ, 点 $\boldsymbol{x}$ でヤコビ行列が連続であることである.

すると, (式 2.16), (式 2.17) より, $\Delta x \to 0$ のとき, $\varepsilon(x, \Delta x)$, $\varepsilon'(x, \Delta x)$ は共に 0 に収束する.

(式 2.18), (式 2.19) を (式 2.14), (式 2.15) を用いて書き換えれば,

$$\begin{cases} \Delta y = \dfrac{dy}{dx}\Delta x - \varepsilon(x, \Delta x)\Delta x & (2.20) \\ \Delta z = \dfrac{dz}{dy}\Delta y - \varepsilon'(x, \Delta x)\Delta y. & (2.21) \end{cases}$$

(式 2.21) に (式 2.20) を代入すると,

$$\Delta z = \frac{dz}{dy}\frac{dy}{dx}\Delta x - \varepsilon(x, \Delta x)\frac{dz}{dy}\Delta x - \varepsilon'(x, \Delta x)\frac{dy}{dx}\Delta x + \varepsilon(x, \Delta x)\varepsilon'(x, \Delta x)\Delta x.$$

したがって,

$$\frac{dz}{dx} = \lim_{\Delta x \to 0}\frac{\Delta z}{\Delta x} = \frac{dz}{dy}\frac{dy}{dx}$$

より, z は x で微分可能で $\dfrac{dz}{dx} = \dfrac{dz}{dy}\dfrac{dy}{dx}$ が成り立つ.

(1) 一変数関数の連鎖律 (別解: 新しい表記の導入)

直前の解答に出てきた $\varepsilon(x, \Delta x)\Delta x$ や $\varepsilon'(x, \Delta x)\Delta x$ を簡単に表す記号を導入する.

0 に収束する変数を**微小数**という. 独立変数のある一定の変動に伴って α も β も微小数で, しかも $\frac{\beta}{\alpha} \to 0$ ならば, β を α よりも**高位の微小数**という. α よりも高位の微小数を一般的に記号 $o(\alpha)$ で表す[*16]. 以下, この表記の置き換え規則を列挙する.

1. $\frac{A}{\alpha} \to 0$ が成立しているときに, この式を $A + o(\alpha) = 0$ に置き換えることができる. 逆に $A + o(\alpha) = 0$ が成立しているときに, この式を $\frac{A}{\alpha} \to 0$ に置き換えることができる.

   $\because$
   $$\frac{A}{\alpha} \to 0 \Longleftrightarrow \frac{-A}{\alpha} \to 0 \Longleftrightarrow -A = o(\alpha) \Longleftrightarrow A + o(\alpha) = 0.$$

2. $o(\alpha) + o(\alpha)$ は $o(\alpha)$ に置き換えることができる.

   $\because$ $o(\alpha) + o(\alpha)$ の一つ目の項を β, 二つ目の項を γ とおくと, $\frac{\beta}{\alpha} \to 0$, $\frac{\gamma}{\alpha} \to 0$ であるから, $\frac{\beta+\gamma}{\alpha} \to 0$. したがって, $\beta + \gamma$ すなわち $o(\alpha) + o(\alpha)$ は $o(\alpha)$ に置き換えることができる.

3. u が有界のとき, $uo(\alpha)$ は $o(\alpha)$ に置き換えることができる.

   $\because$ $uo(\alpha)$ の $o(\alpha)$ を β とおくと, $\frac{\beta}{\alpha} \to 0$ であるから, $\frac{u\beta}{\alpha} \to 0$. したがって, $u\beta$ すなわち $uo(\alpha)$ は $o(\alpha)$ に置き換えることができる.

[*16] この表記はオーダー表記と呼ばれる. o はスモール・オーと読む. $o(\alpha)$ は α よりも早く 0 に収束する項を表しており, 例えば, $o(x^2)$ は x^3 や $4x^4$ などの項をまとめて表している.

4. $o(\alpha)$ が表している項を $\varepsilon\alpha$ に置き換えると $\varepsilon \to 0$. 逆に, $\varepsilon\alpha$ は $\varepsilon \to 0$ のとき, $o(\alpha)$ に置き換えることができる.

   $\because$　$o(\alpha)$ が表している項を β とおくと, $\frac{\beta}{\alpha} \to 0$. β をさらに $\varepsilon\alpha$ に置き換えると $\frac{\beta}{\alpha} = \frac{\varepsilon\alpha}{\alpha} = \varepsilon \to 0$. 逆に, $\varepsilon \to 0$ のとき, $\frac{\varepsilon\alpha}{\alpha} = \varepsilon \to 0$ であるから, $\varepsilon\alpha$ を $o(\alpha)$ に置き換えることができる.

5. u が有界のとき, $o(u\alpha + o(\alpha))$ は $o(\alpha)$ に置き換えることができる. 逆に, $o(\alpha)$ は $o(u\alpha + o(\alpha))$ に置き換えることができる.

   $\because$　$o(u\alpha + o(\alpha))$ が表している項を $\varepsilon(u\alpha + \varepsilon'\alpha)$ とおくと, $\varepsilon \to 0$, $\varepsilon' \to 0$ となる. $\varepsilon(u\alpha + \varepsilon'\alpha) = (\varepsilon u + \varepsilon\varepsilon')\alpha$ は, $\varepsilon u + \varepsilon\varepsilon' \to 0 \cdot u + 0 \cdot 0 = 0$ より, $o(\alpha)$ に置き換えることができる. 逆に $o(\alpha)$ が表している項を $\varepsilon\alpha$ とおくと, $u \to 0$ のとき, $\varepsilon\alpha = \mathrm{sgn}(\varepsilon)\sqrt{|\varepsilon|}(u + (-u + \sqrt{|\varepsilon|}))\alpha$ であり, $\mathrm{sgn}(\varepsilon)\sqrt{|\varepsilon|} \to 0$, $-u + \sqrt{|\varepsilon|} \to 0$ となるから, $o(\alpha)$ を $o(u\alpha + o(\alpha))$ に置き換えることができる. $u \to 0$ でないとき, $\varepsilon\alpha = \frac{\varepsilon}{u}(u + 0)\alpha$ であり, $\frac{\varepsilon}{u} \to 0$, $0 \to 0$ となるから, $o(\alpha)$ を $o(u\alpha + o(\alpha))$ に置き換えることができる.

さて, 直前の解答の (式 2.16), (式 2.17) は, オーダー表記を用いると,

$$\begin{cases} \Delta y = \dfrac{dy}{dx}\Delta x + o(\Delta x) & (2.22) \\ \Delta z = \dfrac{dz}{dy}\Delta y + o(\Delta y) & (2.23) \end{cases}$$

と表すことができる.

(式 2.22) より,

$$\begin{aligned} o(\Delta y) &= o\left(\frac{dy}{dx}\Delta x + o(\Delta x)\right) \\ &= o(\Delta x). \end{aligned}$$

(式 2.23) に (式 2.22) を代入し, この結果 $o(\Delta y) = o(\Delta x)$ を用いると,

$$\begin{aligned} \Delta z &= \frac{dz}{dy}\left(\frac{dy}{dx}\Delta x + o(\Delta x)\right) + o(\Delta x) \\ &= \frac{dz}{dy}\frac{dy}{dx}\Delta x + \frac{dz}{dy}o(\Delta x) + o(\Delta x) \\ &= \frac{dz}{dy}\frac{dy}{dx}\Delta x + o(\Delta x) + o(\Delta x) \\ &= \frac{dz}{dy}\frac{dy}{dx}\Delta x + o(\Delta x). \end{aligned}$$

この式は,

$$\lim_{\Delta x \to 0} \frac{\Delta z}{\Delta x} = \frac{dz}{dy}\frac{dy}{dx}$$

を表している.

(2) 多変数関数の連鎖律

$\boxed{\dfrac{dz}{dx} = \displaystyle\sum_k \frac{\partial z}{\partial y_k}\frac{dy_k}{dx} \text{ が成り立つことをオーダー表記を用いて示す.}}$

x を Δx だけ動かしたときの $\boldsymbol{y}$ の変位を $\Delta \boldsymbol{y}$, z の変位を Δz とする.

z が全微分可能であることと, 全微分係数は $\left(\dfrac{\partial z}{\partial y_j}\right)$ であることから[*17],

$$\begin{aligned}
&\lim_{\|\Delta \boldsymbol{y}\| \to 0} \frac{\|z(\boldsymbol{y}+\Delta \boldsymbol{y}) - z(\boldsymbol{y}) - \left(\frac{\partial z}{\partial y_j}\right)\Delta \boldsymbol{y}\|}{\|\Delta \boldsymbol{y}\|} = 0 \\
\iff \quad & z(\boldsymbol{y}+\Delta \boldsymbol{y}) - z(\boldsymbol{y}) = \left(\frac{\partial z}{\partial y_j}\right)\Delta \boldsymbol{y} + o(\|\Delta \boldsymbol{y}\|) \\
\iff \quad & \Delta z = \left(\frac{\partial z}{\partial y_j}\right)\Delta \boldsymbol{y} + o(\|\Delta \boldsymbol{y}\|). \qquad (2.24)
\end{aligned}$$

y_k が x で微分可能であるから,

$$\Delta y_k = \frac{dy_k}{dx}\Delta x + o(\Delta x). \qquad (2.25)$$

[*17] $\left(\dfrac{\partial z}{\partial y_j}\right)$ は第 j 列目の要素が $\dfrac{\partial z}{\partial y_j}$ であるような行ベクトルを表している.

(式 2.25) より,

$$
\begin{aligned}
{\Delta y_k}^2 &= \left(\frac{dy_k}{dx}\Delta x\right)^2 + 2\frac{dy_k}{dx}\Delta x o(\Delta x) + \{o(\Delta x)\}^2 \\
&= \left(\frac{dy_k}{dx}\Delta x\right)^2 + o(\Delta x^2) \\
\sum_k {\Delta y_k}^2 &= \sum_k \left(\frac{dy_k}{dx}\Delta x\right)^2 + o(\Delta x^2) \\
&= \Delta x^2 \left\{\sum_k \left(\frac{dy_k}{dx}\right)^2 + o(1)\right\} \\
o(\|\Delta \boldsymbol{y}\|) &= o\left(\sqrt{\Delta x^2 \left\{\sum_k \left(\frac{dy_k}{dx}\right)^2 + o(1)\right\}}\right) \\
&= o\left(\Delta x \sqrt{\sum_k \left(\frac{dy_k}{dx}\right)^2 + o(1)}\right) \\
&= o(\Delta x).
\end{aligned}
$$

(式 2.24) に (式 2.25) を代入し, この結果 $o(\|\Delta \boldsymbol{y}\|) = o(\Delta x)$ を用いると,

$$
\begin{aligned}
\Delta z &= \left(\frac{\partial z}{\partial y_j}\right)\left(\frac{dy_i}{dx}\Delta x + o(\Delta x)\right) + o(\Delta x) \\
&= \left(\frac{\partial z}{\partial y_j}\right)\left(\frac{dy_i}{dx}\Delta x\right) + o(\Delta x) \\
&= \left(\sum_k \frac{\partial z}{\partial y_k}\frac{dy_k}{dx}\right)\Delta x + o(\Delta x).
\end{aligned}
$$

したがって, z は x で微分可能であり, $\dfrac{dz}{dx} = \displaystyle\sum_k \frac{\partial z}{\partial y_k}\frac{dy_k}{dx}$ をみたす.

(3) 多変数関数の連鎖律

$\dfrac{\partial z}{\partial x_i} = \displaystyle\sum_k \frac{\partial z}{\partial y_k}\frac{\partial y_k}{x_j}$ が成り立つことをオーダー表記を用いて示す.

$\boldsymbol{x}$ を $\Delta \boldsymbol{x}$ だけ動かしたときの $\boldsymbol{y}$ の変位を $\Delta \boldsymbol{y}$, z の変位を Δz とする.

全問と同様に,

$$
\Delta z = \left(\frac{\partial z}{\partial y_j}\right)\Delta \boldsymbol{y} + o(\|\Delta \boldsymbol{y}\|) \tag{2.26}
$$

が成り立つ. また, $\boldsymbol{y}$ が $\boldsymbol{x}$ で全微分可能であるから, y_k も $\boldsymbol{x}$ で全微分可能であり,

$$
\Delta y_k = \left(\frac{\partial y_k}{\partial x_j}\right)\Delta \boldsymbol{x} + o(\|\Delta \boldsymbol{x}\|) \tag{2.27}
$$

が成り立つ.

(式 2.27) より,

$$
\begin{aligned}
|\Delta y_k|^2 &= \left| \left(\frac{\partial y_k}{\partial x_j} \right) \Delta \boldsymbol{x} + o(\|\Delta \boldsymbol{x}\|) \right|^2 \\
&\leqq \left| \left\| \left(\frac{\partial y_k}{\partial x_j} \right) \right\| \|\Delta x\| + |o(\|\Delta \boldsymbol{x}\|)| \right|^2 \\
&= \left| \left\| \left(\frac{\partial y_k}{\partial x_j} \right) \right\| + |o(1)| \right|^2 \|\Delta \boldsymbol{x}\|^2 \\
\sum_k |\Delta y_k|^2 &\leqq \sum_k \left| \left\| \left(\frac{\partial y_k}{\partial x_j} \right) \right\| + |o(1)| \right|^2 \|\Delta \boldsymbol{x}\|^2 \\
o(\|\Delta \boldsymbol{y}\|) &\subset o(\|\Delta \boldsymbol{x}\|)
\end{aligned}
$$

(途中, 不等号の評価ではシュワルツの不等式 $|\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}| \leqq \|\boldsymbol{a}\|\|\boldsymbol{b}\|$ を用いた. 式 $o(\|\Delta \boldsymbol{y}\|) \subset o(\|\Delta \boldsymbol{x}\|)$ は, $o(\|\Delta \boldsymbol{y}\|)$ の表す項が $o(\|\Delta \boldsymbol{x}\|)$ の表す項に含まれることを意味する.)

(式 2.26) に (式 2.27) を代入し, この結果 $o(\|\Delta \boldsymbol{y}\|) \subset o(\|\Delta \boldsymbol{x}\|)$ を用いると,

$$
\begin{aligned}
\Delta z &= \left(\frac{\partial z}{\partial y_j} \right) \left(\frac{\partial y_i}{\partial x_j} \Delta \boldsymbol{x} + \begin{pmatrix} o(\|\Delta \boldsymbol{x}\|) \\ o(\|\Delta \boldsymbol{x}\|) \\ \vdots \\ o(\|\Delta \boldsymbol{x}\|) \end{pmatrix} \right) + o(\|\Delta \boldsymbol{x}\|) \\
&= \left(\frac{\partial z}{\partial y_j} \right) \left(\frac{\partial y_i}{\partial x_j} \right) \Delta \boldsymbol{x} + o(\|\Delta \boldsymbol{x}\|) \\
&= \left(\sum_k \frac{\partial z}{\partial y_k} \frac{\partial y_k}{\partial x_j} \right) \Delta \boldsymbol{x} + o(\|\Delta \boldsymbol{x}\|).
\end{aligned}
$$

よって, z は $\boldsymbol{x}$ で全微分可能であり, $\dfrac{\partial z}{\partial x_j} = \displaystyle\sum_k \frac{\partial z}{\partial y_k} \frac{\partial y_k}{\partial x_j}$ が成り立つ.

3 (基本公式)

(1)

$$
\begin{aligned}
\frac{\partial (af + bg)}{\partial \boldsymbol{x}} &= \left(\frac{\partial (af + bg)}{\partial x_j} \right) = \left(a \frac{\partial f}{\partial x_j} + b \frac{\partial g}{\partial x_j} \right) \\
&= a \left(\frac{\partial f}{\partial x_j} \right) + b \left(\frac{\partial g}{\partial x_j} \right) = a \frac{\partial f}{\partial \boldsymbol{x}} + b \frac{\partial g}{\partial \boldsymbol{x}}
\end{aligned}
$$

(2)，(3)　略

(4)

$$\begin{aligned}\frac{\partial(A\boldsymbol{f}+B\boldsymbol{g})}{\partial \boldsymbol{x}} &= \left(\frac{\partial(\sum_k A_{ik}f_k + \sum_k B_{ik}g_k)}{\partial x_j}\right) \\ &= \left(\sum_k A_{ik}\frac{\partial f_k}{\partial x_j} + \sum_k B_{ik}\frac{\partial g_k}{\partial x_j}\right) = A\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + B\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\end{aligned}$$

(5)，(6)　略

(7)

$$\begin{aligned}\frac{\partial \boldsymbol{f}^{\mathrm{T}}\boldsymbol{g}}{\partial \boldsymbol{x}} = \left(\frac{\partial \boldsymbol{f}^{\mathrm{T}}\boldsymbol{g}}{\partial x_j}\right) = \left(\frac{\partial \sum_k f_k g_k}{\partial x_j}\right) &= \left(\sum_k \frac{\partial f_k}{\partial x_j}g_k + \sum_k f_k\frac{\partial g_k}{\partial x_j}\right) \\ &= \boldsymbol{g}^{\mathrm{T}}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \boldsymbol{f}^{\mathrm{T}}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\end{aligned}$$

4 (連鎖律をヤコビ行列で表示)

(1)

$$\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} = \left(\frac{\partial z_i}{\partial x_j}\right) = \left(\sum_k \frac{\partial z_i}{\partial y_k}\frac{\partial y_k}{\partial x_j}\right) = \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$$

5 (ベクトルをベクトル微分)

(1)

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = \left(\frac{\partial x_i}{\partial x_j}\right) = (\delta_{ij}) = I$$

(2)

$$\frac{\partial A\boldsymbol{x}}{\partial \boldsymbol{x}} = \left(\frac{\partial (A\boldsymbol{x})_i}{\partial x_j}\right) = \left(\frac{\partial \sum_k A_{ik}x_k}{\partial x_j}\right) = (A_{ij}) = A$$

(3)

$$\begin{aligned}\frac{\partial \left(f(x_1), f(x_2), \ldots, f(x_n)\right)^{\mathrm{T}}}{\partial \boldsymbol{x}} &= \left(\frac{\partial f(x_i)}{\partial x_j}\right) = \left(\frac{\partial f(x_i)}{\partial x_i}\frac{\partial x_i}{\partial x_j}\right) \\ &= (f'(x_i)\delta_{ij}) = \mathrm{diag}\left((f'(x_i))\right)\end{aligned}$$

(4)

$$\frac{\partial \boldsymbol{a} \odot \boldsymbol{x}}{\partial \boldsymbol{x}} = \frac{\partial \,\mathrm{diag}(\boldsymbol{a})\boldsymbol{x}}{\partial \boldsymbol{x}} = \mathrm{diag}(\boldsymbol{a})$$

6 (スカラーをベクトル微分)

(1)

$$\frac{\partial \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}}{\partial \boldsymbol{x}} = \left(\frac{\partial \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}}{\partial x_j}\right) = \left(\frac{\partial \sum_k a_k x_k}{\partial x_j}\right) = (a_j) = \boldsymbol{a}^{\mathrm{T}}$$

(2)

$$\frac{\partial \boldsymbol{x}^{\mathrm{T}}\boldsymbol{a}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{a}^{\mathrm{T}}$$

(3)

$$\frac{\partial \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^{\mathrm{T}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} + \boldsymbol{x}^{\mathrm{T}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} = \boldsymbol{x}^{\mathrm{T}}I + \boldsymbol{x}^{\mathrm{T}}I = 2\boldsymbol{x}^{\mathrm{T}}$$

(4)

$$\begin{aligned}\frac{\partial \boldsymbol{x}^{\mathrm{T}}A\boldsymbol{x}}{\partial \boldsymbol{x}} &= \frac{\partial \boldsymbol{x}^{\mathrm{T}}(A\boldsymbol{x})}{\partial \boldsymbol{x}} = (A\boldsymbol{x})^{\mathrm{T}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{x}} + \boldsymbol{x}^{\mathrm{T}}\frac{\partial A\boldsymbol{x}}{\partial \boldsymbol{x}} \\ &= (A\boldsymbol{x})^{\mathrm{T}}I + \boldsymbol{x}^{\mathrm{T}}A = \boldsymbol{x}^{\mathrm{T}}A^{\mathrm{T}} + \boldsymbol{x}^{\mathrm{T}}A = \boldsymbol{x}^{\mathrm{T}}(A + A^{\mathrm{T}})\end{aligned}$$

(5)

$$\begin{aligned}\frac{\partial(\boldsymbol{x}+\boldsymbol{a})^{\mathrm{T}}(\boldsymbol{x}+\boldsymbol{a})}{\partial \boldsymbol{x}} &= (\boldsymbol{x}+\boldsymbol{a})^{\mathrm{T}}\frac{\partial(\boldsymbol{x}+\boldsymbol{a})}{\partial \boldsymbol{x}} + (\boldsymbol{x}+\boldsymbol{a})^{\mathrm{T}}\frac{\partial(\boldsymbol{x}+\boldsymbol{a})}{\partial \boldsymbol{x}} \\ &= 2(\boldsymbol{x}+\boldsymbol{a})^{\mathrm{T}}\end{aligned}$$

(6)

$$\begin{aligned}\frac{\partial(A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} &= (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}\frac{\partial(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} + (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}\frac{\partial(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} \\ &= 2(A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}A\end{aligned}$$

(7)

$$\begin{aligned}\frac{\partial(A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}C(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} &= \frac{\partial(A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}\{C(A\boldsymbol{x}+\boldsymbol{b})\}}{\partial \boldsymbol{x}} \\ &= \{C(A\boldsymbol{x}+\boldsymbol{b})\}^{\mathrm{T}}\frac{\partial(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} + (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}\frac{\partial C(A\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{x}} \\ &= (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}C^{\mathrm{T}}A + (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}CA \\ &= (A\boldsymbol{x}+\boldsymbol{b})^{\mathrm{T}}(C+C^{\mathrm{T}})A\end{aligned}$$

7 (スカラーを行列微分)

(1)

$$\frac{\partial \operatorname{tr}(X)}{\partial X} = \left(\frac{\partial \sum_k X_{kk}}{\partial X_{ij}}\right) = \left(\sum_k \frac{\partial X_{kk}}{\partial X_{ij}}\right) = \left(\sum_k \delta_{ki}\delta_{kj}\right) = (\delta_{ij}) = I$$

(2)

$$\begin{aligned}\frac{\partial \operatorname{tr}(AX)}{\partial X} &= \left(\frac{\partial \sum_k (XA)_{kk}}{\partial X_{ij}}\right) = \left(\frac{\partial \sum_k \sum_l X_{kl}A_{lk}}{\partial X_{ij}}\right) \\ &= \left(\sum_{k,l} \delta_{ki}\delta l j A_{lk}\right) = (A_{ji}) = A^{\mathrm{T}}\end{aligned}$$

(3)

$$\begin{aligned}\frac{\partial \operatorname{tr}(XA)}{\partial X} &= \left(\frac{\partial \sum_k (AX)_{kk}}{\partial X_{ij}}\right) = \left(\frac{\partial \sum_k \sum_l A_{kl} X_{lk}}{\partial X_{ij}}\right) \\ &= \left(\sum_{k,l} A_{kl}\delta_{li}\delta kj\right) = (A_{ji}) = A^{\mathrm{T}}\end{aligned}$$

(4)

$$\begin{aligned}\frac{\partial \operatorname{tr}(X^{\mathrm{T}}AX)}{\partial X} &= \left(\frac{\partial \sum_k \sum_{l,m} X_{lk}A_{lm}X_{mk}}{\partial X_{ij}}\right) \\ &= \left(\sum_{k,l,m}\left(\frac{\partial X_{lk}}{\partial X_{ij}}A_{lm}X_{mk} + X_{lk}A_{lm}\frac{\partial X_{mk}}{\partial X_{ij}}\right)\right) \\ &= \left(\sum_{k,l,m}(\delta_{li}\delta_{kj}A_{lm}X_{mk} + X_{lk}A_{lm}\delta_{mi}\delta_{kj})\right) \\ &= \left(\sum_m A_{im}X_{mj} + \sum_l X_{lj}A_{li}\right) \\ &= \left(\sum_m A_{im}X_{mj}\right) + \left(\sum_l X_{lj}A_{li}\right) \\ &= AX + A^{\mathrm{T}}X \\ &= (A + A^{\mathrm{T}})X\end{aligned}$$

(5)

$$\frac{\partial f(X)}{\partial X^{\mathrm{T}}} = \left(\frac{\partial f(X)}{\partial X_{ji}}\right) = \left(\frac{\partial f(X)}{\partial X}\right)^{\mathrm{T}}$$

(6)

$$\frac{\partial \det(X)}{\partial X} = \left(\frac{\partial \det(X)}{\partial X_{ij}}\right) = \left(\frac{\partial \sum_{kl}\Delta_{kl}X_{kl}}{\partial X_{ij}}\right) = (\Delta_{ij}) = \det(X)(X^{-1})^{\mathrm{T}}$$

式中の Δ_{ij} は行列 X の ij 余因子を表す.

(7)

$$\begin{aligned}
\frac{\partial \ln\det(X)}{\partial X} &= \left(\frac{\partial \ln\det(X)}{\partial X_{ij}}\right) \\
&= \left(\frac{\partial \ln\det(X)}{\partial \det(X)}\frac{\partial \det(X)}{\partial X_{ij}}\right) \\
&= \frac{1}{\det(X)}\left(\frac{\partial \det(X)}{\partial X_{ij}}\right) \\
&= \frac{1}{\det(X)}\det(X)(X^{-1})^{\mathrm{T}} \\
&= (X^{-1})^{\mathrm{T}}
\end{aligned}$$

(8)

$$\begin{aligned}
\frac{\partial \|AX+B\|_{\mathrm{F}}^2}{\partial X} &= \left(\frac{\partial \sum_{k,l}\left\{\sum_m A_{km}X_{ml}+B_{kl}\right\}^2}{\partial X_{ij}}\right) \\
&= \left(\sum_{k,l} 2\left\{\sum_m A_{km}X_{ml}+B_{kl}\right\}\sum_m A_{km}\delta_{mi}\delta_{lj}\right) \\
&= \left(\sum_k 2\left\{\sum_m A_{km}X_{mj}+B_{kj}\right\}A_{ki}\right) \\
&= \left(\sum_k 2A_{ki}\left\{\sum_m A_{km}X_{mj}+B_{kj}\right\}\right) \\
&= 2A^{\mathrm{T}}(AX+B)
\end{aligned}$$

(8) 別解

$\mathrm{tr}(X^{\mathrm{T}}X) = \sum_k (X^{\mathrm{T}}X)_{kk} = \sum_k \sum_l X_{lk}X_{lk} = \sum_{k,l} {X_{lk}}^2 = \|X\|_{\mathrm{F}}^2$ を利用すると,

$$\begin{aligned}
\frac{\partial \|AX+B\|_{\mathrm{F}}^2}{\partial X} &= \frac{\partial \,\mathrm{tr}\left((AX+B)^{\mathrm{T}}(AX+B)\right)}{\partial X} \\
&= \frac{\partial \,\mathrm{tr}\left((X^{\mathrm{T}}A^{\mathrm{T}}+B^{\mathrm{T}})(AX+B)\right)}{\partial X} \\
&= \frac{\partial \,\mathrm{tr}\left(X^{\mathrm{T}}A^{\mathrm{T}}AX + X^{\mathrm{T}}A^{\mathrm{T}}B + B^{\mathrm{T}}AX + B^{\mathrm{T}}B\right)}{\partial X} \\
&= (A^{\mathrm{T}}A + (A^{\mathrm{T}}A)^{\mathrm{T}})X + A^{\mathrm{T}}B + A^{\mathrm{T}}B + O \\
&= 2A^{\mathrm{T}}(AX+B)
\end{aligned}$$

(9)

$$\begin{aligned}
\frac{\partial \|XA+B\|_{\mathrm{F}}}{\partial X} &= \frac{\partial \|A^{\mathrm{T}}X^{\mathrm{T}}+B^{\mathrm{T}}\|_{\mathrm{F}}^2}{\partial X} \\
&= \left(\frac{\partial \|A^{\mathrm{T}}X^{\mathrm{T}}+B^{\mathrm{T}}\|_{\mathrm{F}}^2}{\partial X^{\mathrm{T}}}\right)^{\mathrm{T}} \\
&= \left(2A(A^{\mathrm{T}}X^{\mathrm{T}}+B^{\mathrm{T}})\right)^{\mathrm{T}} \\
&= 2(XA+B)A^{\mathrm{T}}
\end{aligned}$$

8 (単層ニューラルネットワークの勾配降下法を用いた学習)

(1)

$$\frac{\partial L}{\partial W} = \left(\sum_k \frac{\partial L}{\partial y_k}\frac{\partial y_k}{\partial W_{ij}}\right) = \left(\sum_k \frac{\partial L}{\partial y_k}\frac{\partial \sum_l W_{kl}x_l + b_k}{\partial W_{ij}}\right) = \left(\frac{\partial L}{\partial y_i}x_j\right) = \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)^{\mathrm{T}}\boldsymbol{x}^{\mathrm{T}}$$

(2)

$$\frac{\partial L}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{y}}\frac{\partial (W\boldsymbol{x}+\boldsymbol{b})}{\partial \boldsymbol{b}} = \frac{\partial L}{\partial \boldsymbol{y}}I = \frac{\partial L}{\partial \boldsymbol{y}}$$

9 (アフィンレイヤの誤差逆伝播)

$$\frac{\partial L}{\partial \boldsymbol{y}} = \frac{\partial L}{\partial \boldsymbol{y}'}\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{p}}\frac{\partial \boldsymbol{p}}{\partial \boldsymbol{y}} = \boldsymbol{e}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}\frac{\partial (W\boldsymbol{y}+b)}{\partial \boldsymbol{y}} = \boldsymbol{e}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}W$$

10 (ソフトマックス関数＋クロスエントロピー誤差の誤差逆伝播)

(連鎖律を用いて解く)

まず, $\dfrac{\partial y_k}{\partial a_j}$ を求める.

$$\frac{\partial y_k}{\partial a_j} = \frac{\partial}{\partial a_j}\frac{\exp a_k}{\sum_l \exp a_l}. \tag{2.28}$$

(式 2.28) は $k = j$ のとき,

$$\begin{aligned}(\text{式 } 2.28) &= \frac{(\exp a_k)(\sum_l \exp a_l) - (\exp a_k)(\exp a_k)}{(\sum_l \exp a_l)^2} \\ &= y_k - {y_k}^2\end{aligned}$$

となり, $k \neq j$ のとき,

$$\begin{aligned}(\text{式 } 2.28) &= \exp a_k \cdot \frac{\exp a_j}{(\sum_l \exp a_l)^2} \\ &= -y_k y_j\end{aligned}$$

となる.

次に, $\frac{\partial L}{\partial y_k}$ を求めると,

$$\frac{\partial L}{\partial y_k} = \frac{\partial\left(-\sum_l t_l \ln y_l\right)}{\partial y_k} = -\frac{t_k}{y_k}.$$

以上を踏まえて計算すると,

$$\begin{aligned}\frac{\partial L}{\partial a_j} &= \sum_k \frac{\partial L}{\partial y_k}\frac{\partial y_k}{\partial a_j} \\ &= -\frac{t_j}{y_j}(y_j - {y_j}^2) + \sum_{k\neq j}\frac{t_k}{y_j}y_k y_j \\ &= -t_j + t_j y_j + \sum_k t_k y_k - t_j y_j \\ &= -t_j + \left(\sum_k t_k\right) y_k \\ &= -t_j + y_j\end{aligned}$$

が得られる.

(別解: 代入してから解く)

$$
\begin{aligned}
L &= -\sum_k t_k \ln y_k \\
&= -\sum_k t_k \ln \frac{\exp a_k}{\sum_l \exp a_l} \\
&= -\sum_k t_k \left(a_k - \ln \sum_l \exp a_l \right) \\
&= -\sum_k t_k a_k + \left(\sum_k t_k \right) \ln \sum_l \exp a_l \\
&= -\sum_k t_k a_k + \ln \sum_l \exp a_l \\
\frac{\partial L}{\partial a_j} &= -t_j + \frac{\exp a_j}{\sum_l \exp a_l} \\
&= -t_j + y_j
\end{aligned}
$$

11 (リカレントニューラルネットワークの誤差逆伝播)

$$
\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{p}(t-1)} &= \frac{\partial L}{\partial \boldsymbol{p}(t)} \frac{\partial \boldsymbol{p}(t)}{\partial \boldsymbol{h}(t-1)} \frac{\partial \boldsymbol{h}(t-1)}{\partial \boldsymbol{p}(t-1)} \\
&= \boldsymbol{e}(t) \frac{\partial \left(U\boldsymbol{x}(t) + W\boldsymbol{h}(t-1) + \boldsymbol{b} \right)}{\partial \boldsymbol{h}(t-1)} \frac{\partial (f(p_i(t-1)))}{\partial \boldsymbol{p}(t-1)} \\
&= \boldsymbol{e}(t) W \operatorname{diag} \left((f(p_i(t-1))) \right) \\
&= \boldsymbol{e}(t) W \odot (f(p_j(t-1)))
\end{aligned}
$$

12 (京大情報学研究科 修士課程 知能情報学専攻 入学者選抜試験 2018 年実施 T-3)

設問 5 の $\frac{\partial L}{\partial \boldsymbol{w}}$ の形から, この問題での勾配は本章の定義とは異なり列ベクトルであることが分かる. そのため, この問題の解答では $\frac{\partial \boldsymbol{x}^{\mathrm{T}} \boldsymbol{a}}{\partial \boldsymbol{x}} = \boldsymbol{a}$ となる (覚えるべき公式はこの一つのみである. それ以上の複雑な計算は公式を覚えて使うよりも添え字で計算するほうが間違いがない.).

設問 1

$L_1(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i); \boldsymbol{w})^2$ とすれば, $L_1(\boldsymbol{w})$ は極値で最小値をとるので, $\hat{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathbb{R}^d}{\operatorname{argmin}} \ L_1(\boldsymbol{w})$ は, $\left. \frac{\partial L_1(\boldsymbol{w})}{\partial \boldsymbol{w}} \right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} = \boldsymbol{0}$ を満たす. したがって, $\boldsymbol{y} =$

$[y_1, y_2, \ldots, y_n]^{\mathrm{T}}$ とすると,

$$\begin{aligned} L_1(\boldsymbol{w}) &= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)^{\mathrm{T}} \\ &= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)(\boldsymbol{y} - X^{\mathrm{T}}\boldsymbol{w}) \\ &= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}X^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}X\boldsymbol{y} + \boldsymbol{w}^{\mathrm{T}}XX^{\mathrm{T}}\boldsymbol{w}) \end{aligned}$$

$$\left.\frac{\partial L_1(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\hat{\boldsymbol{w}}} = -X\boldsymbol{y} + XX^{\mathrm{T}}\hat{\boldsymbol{w}} = \boldsymbol{0}$$

$$XX^{\mathrm{T}}\hat{\boldsymbol{w}} = X\boldsymbol{y}$$

ここで, 行列 XX^{T} はランク d の正則行列であることを示す. d 行 n 列の行列 X のランクが d であるので, グラム・シュミットの直交化により, n 次元の行ベクトルからなる正規直交基底 $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d$ を用いて, $X = X' \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \vdots \\ \boldsymbol{u}_d \end{pmatrix} = X'U$ と分解することができる. ここで X' は d 行 d 列の正方行列であり, 正則である. U は正規直交基底からなる d 行 n 列の行列である. この分解を用いると, UU^{T} が d 次の単位行列になることから, $XX^{\mathrm{T}} = X'UU^{\mathrm{T}}X'^{\mathrm{T}} = X'X'^{\mathrm{T}}$ となり, X' は正則なので XX^{T} も正則である.

以上より, 行列 XX^{T} は正則行列なので, $\hat{\boldsymbol{w}} = \left(XX^{\mathrm{T}}\right)^{-1}X\boldsymbol{y}$ と表すことができる.

設問 2

パラメータ $\boldsymbol{w}$ のとき, 入力 $\boldsymbol{x}_i$ に対する出力 y'_i の確率密度関数 $g(y'_i; \boldsymbol{x}_i, \boldsymbol{w})$ は次のようになる.

$$g(y'_i; \boldsymbol{x}_i, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{(y'_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2}{2}\right\}$$

それぞれの学習データ $(\boldsymbol{x}_i, y_i)$ は独立なので, $y'_i = y_i$ $(i = 1, 2, \ldots, n)$ となる尤度関数 L_2 は,

$$L_2(y_1, y_2, \ldots, y_n, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n; \boldsymbol{w}) = \prod_{i=1}^{n} g(y_i; \boldsymbol{x}_i, \boldsymbol{w})$$

と定義することができる. すると, $\frac{\partial \ln L_2}{\partial \boldsymbol{w}}$ は, 設問 1 の $\frac{\partial L_1}{\partial \boldsymbol{w}}$ を -1 倍したものと等しくなる. したがって, $\boldsymbol{w}$ の最尤推定量 $\hat{\boldsymbol{w}}$ は, 設問 1 と同じく, $\hat{\boldsymbol{w}} = \left(XX^{\mathrm{T}}\right)^{-1}X\boldsymbol{y}$ と表すことができる.

設問３

$\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^{\mathrm{T}}$, $\boldsymbol{r} = [r_1, r_2, \ldots, r_n]^{\mathrm{T}}$ とする. 損失 L_3 を

$$L_3 = \frac{1}{2}\sum_{i=1}^{n} r_i(y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

と定義すると,

$$\begin{aligned}
L_3 &= \frac{1}{2}\sum_{i=1}^{n} r_i(y_i - f(\boldsymbol{x}_i; \boldsymbol{w}))^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 \\
&= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)\mathrm{diag}(\boldsymbol{r})(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)^{\mathrm{T}} + \frac{\lambda}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \\
&= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}} - \boldsymbol{w}^{\mathrm{T}}X)\mathrm{diag}(\boldsymbol{r})(\boldsymbol{y} - X^{\mathrm{T}}\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \\
&= \frac{1}{2}(\boldsymbol{y}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{r})\boldsymbol{y} - \boldsymbol{y}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}X\mathrm{diag}(\boldsymbol{r})\boldsymbol{y} + \boldsymbol{w}^{\mathrm{T}}X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}}\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L_3}{\partial \boldsymbol{w}} &= -X\mathrm{diag}(\boldsymbol{r})\boldsymbol{y} + X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}}\boldsymbol{w} + \lambda\boldsymbol{w} \\
&= (X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I)\boldsymbol{w} - X\mathrm{diag}(\boldsymbol{r})\boldsymbol{y}
\end{aligned}$$

となる. ただし, 式中の I は d 次の単位行列である.

ここで, 行列 $X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I$ は正則行列であることを示す. 行列 $X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}}$ は対称行列であり正定値行列であるので直交行列 Q と対角成分がすべて正である対角行列 Λ を用いて $Q\Lambda Q^{\mathrm{T}}$ と固有値分解することができる. 行列 $\Lambda + \lambda I$ を考えると, 行列 $\Lambda + \lambda I$ の対角成分は全て正であり零でないため, 対角成分がこの対角成分の逆数であるような新たな対角行列 $R = (\Lambda + \lambda I)^{-1}$ を作ることができる. このように定めた直交行列 Q と対角行列 R を用いれば, $(X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I)(QRQ^{\mathrm{T}}) = Q(\Lambda + \lambda I)Q^{\mathrm{T}}QRQ^{\mathrm{T}} = I$ となるので, $X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I$ は逆行列 QRQ^{T} を持つ.

以上より, $\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathbb{R}^d}{\mathrm{argmin}}\ L_3$ は, $X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I$ が正則なので,

$$\hat{\boldsymbol{w}} = (X\mathrm{diag}(\boldsymbol{r})X^{\mathrm{T}} + \lambda I)^{-1}X\mathrm{diag}(\boldsymbol{r})\boldsymbol{y}$$

と表すことができる.

設問 4

出力が 2 値 $\{0,1\}$ であるので, $y=1$ でなければ必ず $y=0$ が出力される. したがって $y=0$ の事後確率の確率モデルは,

$$\begin{aligned} p(y=0|\boldsymbol{x};\boldsymbol{w}) &= 1-p(y=1|\boldsymbol{x};\boldsymbol{w}) \\ &= 1-\frac{1}{1-\exp(-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})} \\ &= \frac{\exp(-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})}{1+\exp(-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})} \end{aligned}$$

と表すことができる.

設問 5

$\phi(z)=\frac{1}{1+\exp(-z)}$ とすると,

$$\begin{aligned} \frac{d\phi(z)}{dz} &= \frac{+\exp(-z)}{(1+\exp(-z))^2} \\ &= \frac{1}{1+\exp(-z)}\left(1-\frac{1}{1+\exp(-z)}\right) \\ &= \phi(z)(1-\phi(z)) \end{aligned}$$

となる. さらに $\phi_i=\phi(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i)=p(y=1|\boldsymbol{x}_i,\boldsymbol{w}) \quad (i=1,2,\ldots,n)$ とすると, 対数尤度関数 $L(\boldsymbol{w})$ は,

$$L(\boldsymbol{w})=\sum_{i=1}^{n}\left(y_i\ln\phi_i+(1-y_i)\ln(1-\phi_i)\right)$$

で与えられるので, その勾配は,

$$\begin{aligned} \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \sum_{i=1}^{n}\left(y_i\frac{d\ln\phi_i}{d\phi_i}\frac{d\phi_i}{d\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}\frac{\partial \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}{\partial \boldsymbol{w}}+(1-y_i)\frac{d\ln(1-\phi_i)}{d\phi_i}\frac{d\phi_i}{d\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}\frac{\partial \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i}{\partial \boldsymbol{w}}\right) \\ &= \sum_{i=1}^{n}\left(y_i\frac{1}{\phi_i}\phi_i(1-\phi_i)\boldsymbol{x}_i-(1-y_i)\frac{1}{1-\phi_i}\phi_i(1-\phi_i)\boldsymbol{x}_i\right) \\ &= \sum_{i=1}^{n}\left(y_i(1-\phi_i)\boldsymbol{x}_i-(1-y_i)\phi_i\boldsymbol{x}_i\right) \\ &= \sum_{i=1}^{n}\left(y_i-\phi_i\right)\boldsymbol{x}_i \\ &= \sum_{i=1}^{n}\left(y_i-p(y=1|\boldsymbol{x}_i,\boldsymbol{w})\right)\boldsymbol{x}_i \end{aligned}$$

と表すことができる.

第 3 章

脳から学ぶ人工知能

3.1　はじめに

本会誌の第 1 章で紹介した**畳み込みニューラルネットワーク**（convolutional neural network、**略して** CNN）と呼ばれる手法は、自動運転や音声認識をはじめとした最先端の技術の数々に採用されている、今最もホットな技術の一つです。例えば自動運転では道路の白線を認識するために、顔認識では画像から顔を検出して切り出すために利用され、他にも翻訳アプリや音声認識でも利用されています。

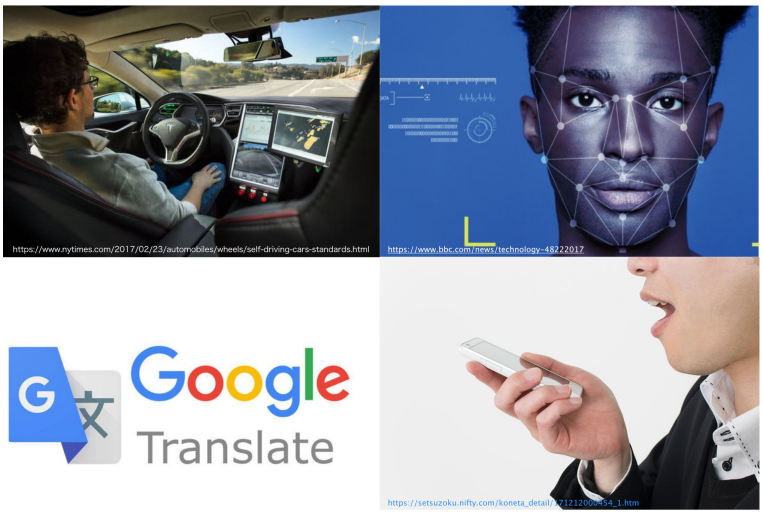


図 3.1 CNN の応用例

このように産業分野で広く使われている畳み込みニューラルネットワーク（CNN）ですが、元々は誰がどのように発明したのでしょうか？　実は、CNN の研究は脳科学に端を発しています。もともとニューラルネットワーク自体脳のモデルとして作られたものであ

り、CNN はさらに脳の視覚野（目で見た情報を処理する脳部位）をモデルとして開発されました。そして、脳のモデルから CNN を開発するまでの道のりには、二人の日本人研究者が大きく関わっています。CNN の構造を設計した**福島邦彦**先生と、CNN の学習アルゴリズムを考案した**甘利俊一**先生です。特に福島先生は CNN の開発に重要な役割を果たした研究者として、世界中で広く知られています。この章では、脳科学の研究が CNN の開発につながるまでの歴史と、その中で大きな役割を果たした二人の研究を紹介します。

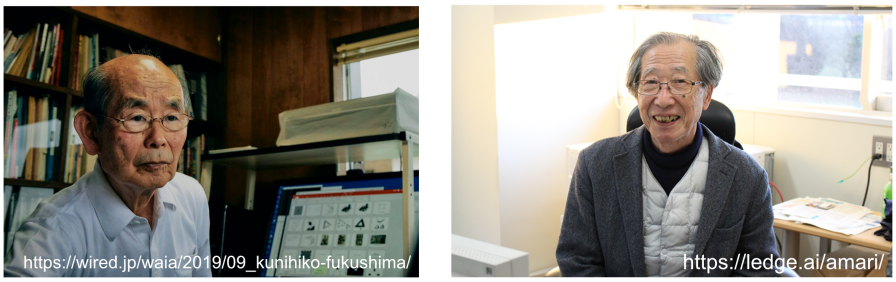


図 3.2 左：福島邦彦先生　右：甘利俊一先生

視覚野の神経細胞とそのモデル（Hubel & Wiesel）

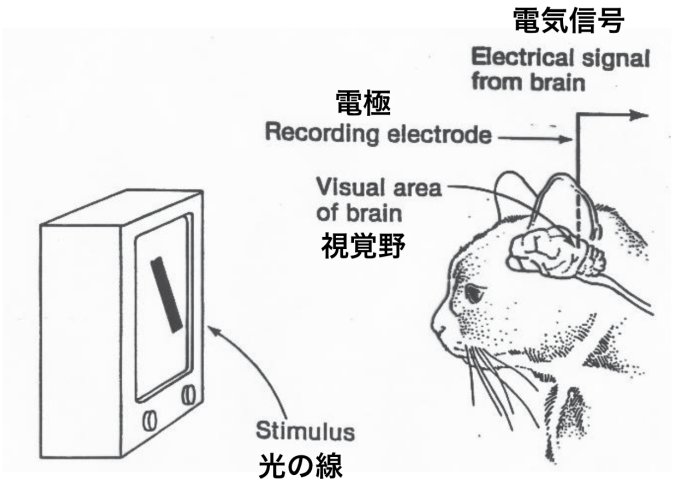
CNN の歴史を説明する前に、まずはその元となった脳科学の研究を紹介します。Hubel と Wiesel はハーバード大学の研究者で、「神経生理学者」と呼ばれる細胞や組織の機能を明らかにすることを目指す研究者でした。二人は脳の視覚野にある神経細胞の機能を調べるため、図 3 のような実験 [7] を行いました。

この実験では、まず猫の頭を固定して、視覚野に電極を挿しておきます。神経細胞は電気で情報を伝達するので、電極から神経細胞の活動を記録できます。次に、猫の正面に上図のような光の線を見せます。そして、光の線を上下左右に動かしていきます。

Hubel と Wiesel は、視覚野の神経細胞の中に、「ある特定の部分に特定の角度の線が見えたとき」のみに反応する細胞があることを発見しました。視覚野には線の位置と角度を認識する細胞が存在したということです。私たちが実際に物体の角度を認識できる以上、脳の中に角度を認識できる細胞があることは不思議ではないかもしれません。しかし、実際にそれを発見したのは Hubel と Wiesel が最初でした。

Hubel と Wiesel はさらにもう一種類の細胞を発見しました。その細胞は「どの位置にあっても、特定の角度の線が見えたとき」に反応する細胞でした。つまり、位置にはよらず角度のみに反応する細胞です。

前者の特定の位置と特定の角度の組み合わせに反応する細胞を**単純型細胞**、後者の位置にはよらず角度のみに反応する細胞を**複雑型細胞**と呼びます。角度だけに反応する細胞の



https://goodpsychology.wordpress.com/2013/03/13/235/より改変

図 3.3 Hubel と Wiesel による実験

方が単純のようにも思いますが、より抽象的な刺激に反応する、という点で後者の方を複雑型と呼んでいます。

さらに、Hubel と Wiesel はこの二つの細胞を数式で表しました。数式は難しいので省きますが、二つの細胞は下図のように表されます。

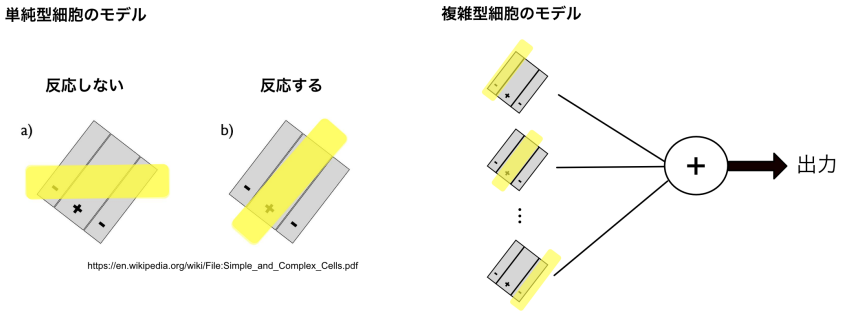


図 3.4 左：単純型細胞のモデル　右：複雑型細胞のモデル

単純型細胞は真ん中の + の両端を - で挟んだフィルターとして表します。このフィルターと同じ角度に光の線がくれば検出され、異なる角度にくれば検出されません。そして、このフィルターが視野を埋め尽くすように並んでいる、としました。複雑型細胞は、単純型細胞の足し合わせで表せます。視野のどこかで傾きが検出されれば、それが足し合

わされ、複雑型細胞も反応します。このようにして位置によらず角度だけに反応する細胞を作ることができます。

以上が 1959 年に行われた Hubel と Wiesel の研究です。この実験により、視覚野の研究はその後急速に進んでいきます。二人はこの業績で 1981 年にノーベル医学・生理賞を受賞しており、世界中の脳科学者に広く知られています。

3.2　ネオコグニトロン（福島）

上記の Hubel と Wiesel の研究を見て、その仕組みを人工知能に取り入れようとしたのが福島邦彦先生でした。福島先生は、上記の単純型細胞と複雑型細胞のモデルとを交互に繰り返すことで画像認識をおこなうモデルを開発しました。今では単純型細胞は**畳み込み層**、複雑型細胞は **プーリング層** と呼ばれています。これがのちに CNN の前身として知られるようになった **ネオコグニトロン** [8] です。

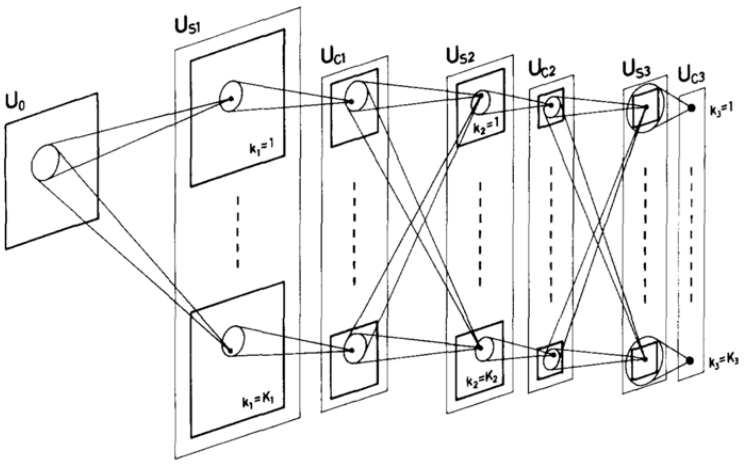


図 3.5 ネオコグニトロンの概略図。図中の U_s が単純型細胞に、U_C が複雑型細胞に相当する。(Fukushima, 1986) より引用

一般的に、人工知能のアルゴリズムには二つ設計する必要があるものがあります。一つ目は**モデルの構造**、二つ目は**学習アルゴリズム**です。学習アルゴリズムとは、作成した人工知能が実際にデータから学習するための仕組みです。ネオコグニトロンは、モデルの構造に上で紹介した畳み込みとプーリングの繰り返しを用いる一方、学習アルゴリズムには AiS（Add-if-Silent）と呼ばれる仕組みを採用しました。AiS では「前の層のプーリングが反応したにもかかわらず今の層の畳み込みが反応しないとき、そこに畳み込みをひとつ加える」という方法で学習します。入力画像セットを一回入れれば学習が終了するので、

非常に高速に学習できます。　しかしながら、この学習アルゴリズムはそれほど強力ではありませんでした。その理由の一つは AiS が「教師なし学習」であることでしょう。一般的に人工知能の学習アルゴリズムには「教師あり学習」と「教師無し学習」が存在します（第 1 章参照）。教師あり学習では人工知能に入力データとそれに対する答えを用意します。画像認識を行う場合であれば、入力の画像とその物体のカテゴリーの両方を人工知能に与えます。一方で教師無し学習は画像だけを人工知能に与えます。教師あり学習の方がうまく学習ができそうな気がしますが、当時多層のネットワークに対応できる教師あり学習のアルゴリズムは知られていませんでした。また、当時はいちいち人工知能の出力と答えを照合するのは効率的ではないという考えもあり、教師あり学習の方を採用したそうです [10]。次節では、より強力な多層のネットワークに適用できる教師あり学習のアルゴリズムを紹介します。

コラム：NHK 放送技術研究所

福島先生はネオコグニトロンの開発当時、NHK 放送技術研究所に所属されていました。現代の人工知能の基礎技術が日本の、しかも人工知能とは全く関係なさそうな NHK で開発されていたというのは驚きです。当時の NHK 研究所はどのような場所だったのでしょうか？　福島先生は Wired のインタビュー [10] の中で次のように話しています。

> ... そもそも研究所のあり方が、当時においてとても斬新でした。生理学、心理学、工学の人々を集めて研究所を立ち上げた樋渡涓二さんという、わたしたちのボスがいまして、研究のことを非常によくわかっている方だったんです。「きみたちは研究をする上で、NHK のことなんか忘れろ」なんて言ってくれる人だったんですよ（笑）学界に寄与できることをやれば、それは巡り巡って NHK に返ってくる。だから細かいことを考えるな、ということを言われたわけなんですね。彼は非常に先見の明があったと思います。何の役に立つのかを特に聞かれず、自由に研究させてもらいました。...

当時の NHK の研究所はだいぶ自由な発想で研究ができる場であったようです。本当に世の中の役に立つ技術は自由な環境の中で生まれるのだ、ということを再確認するようなエピソードです。

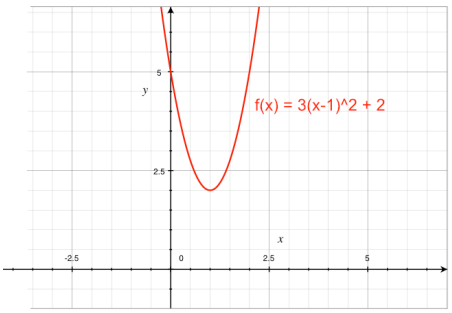
コラム：ネオコグニトロンと ReLU 関数

福島先生の考案したネオコグニトロンはそのまま使われることはないものの、一部のアイデアは CNN の性能向上に役立つことが再発見されています。アイデアの一つには、ReLU 関数があります。CNN が発明された当時、活性化関数には sigmoid 関数または tanh を使うことが普通でした。おそらく、微分がきれいな形になるからだったのでしょう。しかし、その後の研究で ReLU の方が計算も早く、勾配消失が起こりにくいことがわかりました。ネオコグニトロンに ReLU　が使われているのは偶然でしょうが、福島先生の先見の明があったのだとも言えるかもしれません。

3.3　誤差逆伝播法（Rumelhart、甘利）

1980 年代当時、二層のニューラルネットワークに適用できる教師あり学習のアルゴリズムは既に知られていました。これはデルタルールといい、一言で言えば微分を使って出力誤差を最小にするアルゴリズムでした。高校生の時に微分を習った経験があれば、次のような問題を解かされたことでしょう。

問題： 次の関数の最小値を求めよ。$f(x) = 3(x-1)^2 + 2$



この場合すぐに $x = 1$ が答えだとわかりますが、微分を使って解く場合は次のように解くと思います

解答：　を x で微分すると、$f'(x) = 6(x-1)$。$f'(x) = 0$ を解いて $x = 1$。$f''(x) = 6 > 0$ なので $x = 1$ が最小値

このような問題では、微分して 0 とおけば解けます。ただ、関数の全体像がわかっていない場合、どう解けばいいでしょうか。その場合は関数の最小値はわかりませんが、関数の値を小さくすることはできます。図のように、微分を計算した後に微分の符号と逆向きに x を動かします。そして微分が 0 になればそこが最小値の候補です。微分は局所的な

傾きなので、関数の全体像がわからなくても計算することが可能です。

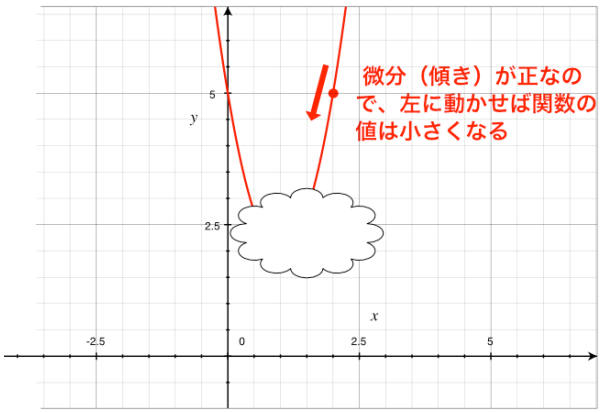


図 3.6

ニューラルネットワークに応用するためには上記の x をネットワークのパラメータに変更し、$f(x)$ を出力誤差の関数に変更します。上の問題と同じようにして x で微分して逆向きに動かすことで出力誤差を小さくでき、性能の良い人工知能が作れる、というのがデルタルールの基本的な考え方です。このデルタルールの前身は Rescorla と Wagner という二人の心理学者によって作られました [11]。

このデルタルールを拡張すれば、3 層以上のネットワークにも同じような方法で誤差関数を小さくできます。そのために、これも高校生の時に習った合成関数の微分の公式を用います。ニューラルネットワークを多層にするということは、関数を複数回合成するのと同じことです。そのため合成関数の微分の公式を用いれば簡単に微分を計算できます（より詳細な説明は、会誌の第 1 章と第 2 章を参照してください）。この合成関数の微分を用いるアイデアは誤差逆伝播法（Back propagation）といい、Rumelhart らが 1986 年の論文 [14] で提案したことが広く知られています。

しかしながら、似たアイデアは以前にも何度か提案されていたようです。その中でおそらく最も早くこのアイデアを提案したのは、20 年以上前の 1967 年に出版された **甘利俊一**先生（当時、東京大学所属）の論文 [13] でした。なぜ、同じアイデアを 20 年も早く出したのに、甘利先生の論文は有名にならなかったのでしょうか？　アイデアを出すのが早すぎたこと、1970 年になると人工知能ブームが一旦終わってしまったこと、そして個人的な感想ですが、甘利先生の論文はちょっと難しすぎたことが原因として挙げられると思います。

このように、残念にも功績があまり認知されなかった甘利先生ですが、誤差逆伝播法の発見の他にも数多くの業績を残されています（脳の数理的理論の考案、情報幾何学

（information geometry）の創始など）。これらの功績が認められ、甘利先生は今年度の文化勲章を受賞されました。

3.4　CNN の発明（Lecun）

福島先生のネオコグニトロンの構造を採用しつつ Rumelhart や甘利先生の発明した誤差逆伝播法を学習アルゴリズムに使用することでより効率的な人工知能が作られました。これは当時 AT&T ベル研究所に所属していた Lucun によって発明され、LeNet と名付けられました [15]。

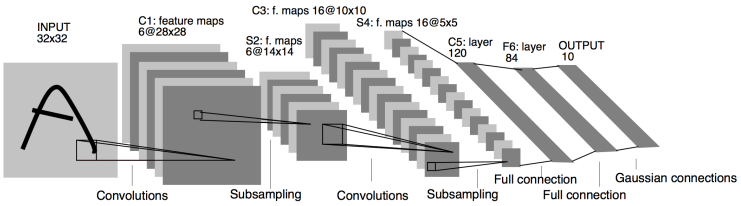


図 3.7 LeNet の概略図　(Lecun, Bottou, Bengio, & Haffner, 1998) より引用

LeNet 以降 AlexNet, VGG, GoogLeNet, ResNet など様々な CNN が開発されましたが、これら全ての CNN も 基本は LeNet と同じ仕組みです。つまり、連続した畳み込み層を持ち、誤差逆伝播法によって学習します。

Lecun はこの業績によって、コンピューター科学で最も権威のあるチューリング賞を受賞しました。

3.5　脳と AI のこれから

ここまで、脳科学の実験が CNN の発明につながるまでの歴史を紹介しました。その後人工知能技術はしばらくの間脳科学とは離れて発展することになりますが、現在再び人工知能と脳科学の関係が見直されています。最後に、いくつかの興味深いトピックを紹介します。

1. CNN と脳の類似性

脳から着想を得て作られた CNN ですが、そうして作られた CNN は脳と似ているのでしょうか？ Yamins らは 2014 年の研究 [16] で、画像をサルに見せたときの視覚野の脳活動が、同じ画像を CNN に入力したときの中間層の値から線形回帰によって予測できることを示しました。

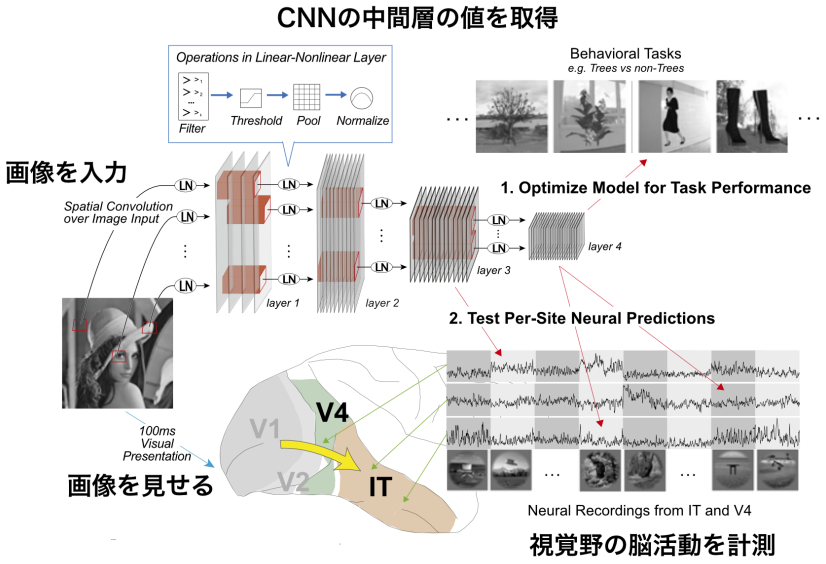


図 3.8 Yamins らの実験の概略図　(Yamins et al., 2014) より改変

CNN から脳が予測できるということは、CNN と脳が「似た情報を持っている」ことを示しています。もし、CNN と脳で全く違う情報処理がされていたら、CNN の中間層の値と脳活動がもつ情報は異なるものになるはずです。その場合、CNN から脳の予測はほとんどできないでしょう。Yamins らの実験では、予測ができることを示し、CNN と脳が「似ている」ことを示したのです。

興味深い点は、この CNN 自体は脳活動を予測するために訓練されたものではなく、単に画像認識用に訓練されていたという点です。もともと脳をまねて作られた CNN ですが、CNN の構造は脳と比べてかなり単純化されていますし、誤差逆伝播法は脳からヒントを得たわけではありません。にもかかわらず、CNN が脳のモデルに使えるというのは驚くべきことです。現在でも CNN は脳のモデルとして活発に利用・研究されています。

一つ例として、CNN を脳のモデルとして使うことで実現した面白い技術を紹介します。京都大学/ATR の神谷教授らのグループは CNN を用いることで、ヒトが思い浮かべている画像を脳活動から読み出す技術の開発に成功しました [19]。

この研究では Yamins らとは逆に、脳活動から CNN の中間層の値を予測しました。脳活動から CNN を値を予測することは、脳活動を CNN の値に「変換する」することを可能にします。いったん CNN の値に変換されれば、今度は CNN の中間層の値がその値になるような入力画像を作成することができます。このようにして脳活動→ CNN →画像と順を追って変換することで、ヒトが思い浮かべている画像を脳から読み出すことができるのです。

図 9 が脳から読み出された画像です。画像を見ている時の脳活動から再構成された画像（左）はかなりはっきりと輪郭が見て取れます。画像を思い描いている時の脳活動から再構成された画像（右）はぼんやりですが、大体の形は再現されているのがわかります。

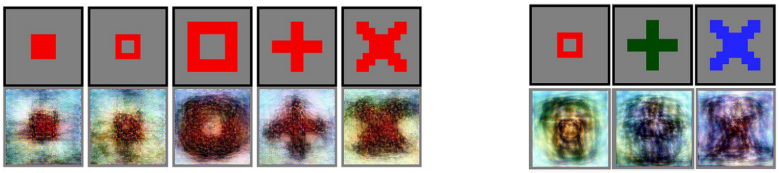


図 3.9 上段が元の画像、下段が脳活動から再構成された画像。左：見ている時の脳活動から　右：思い浮かべている時の脳活動から　(Shen el al., 2019) より改変

ちなみにこの研究で脳活動を測るために使った MRI は磁気によって脳活動を計測するので、電極などと違って脳に直接触れることはありません。脳に指一本触れることなく人間が見ているものを当てる様子は、昔の人が見たら魔法のようだと思いことでしょう。

2. 脳から着想を得たニューラルネットワークの改善手法

初期の CNN は脳からヒントを得ましたが、それ以外でも脳は人工知能を発展させるヒントを与えています。例えば、人間がもつ「有用な情報だけに集中する」という注意のメカニズムは ニューラルネットワーク 以前にも多数用いられてきました。（実は福島先生もネオコグニトロンの発展系として注意のメカニズムを導入した物を考案しています）。その中でも Attention（英語で注意の意）[21] と呼ばれるメカニズムは性能向上に大きく役立つとして頻繁に使用されています。

Attention の威力が知られるようになったきっかけは、seq2seq [20] という翻訳などに使われるモデルへの適用でした。このモデルは元の言語の文章を一単語ずつ受け取ったのち、その文章の翻訳を生成します。

seq2seq もニューラルネットワークの一種ですが、 CNN とは違い畳み込み層ではなく LSTM と呼ばれる仕組みを使います。しかし、誤差逆伝播法で学習する点は同じです。図 10 を見るとわかるように、seq2seq は元の文章を全て入力し終わってから翻訳を開始します。これが seq2seq の大きな弱点でした。文章が長すぎると途中の情報が失われてしまうからです。一方人間が翻訳するときは、文章全体を見つつもひとつ一つの単語やフレーズに注目しながら翻訳するため、長い文章でも正確に翻訳できます。Attention はまさにその仕組みをニューラルネットワークに持たせた物です。

Attention を導入するためには、元の文章の単語を入力したときの出力をそれぞれとっておき、全単語を入力し終わったのちに重み付き和 (図中の h）を求めます。そして翻訳

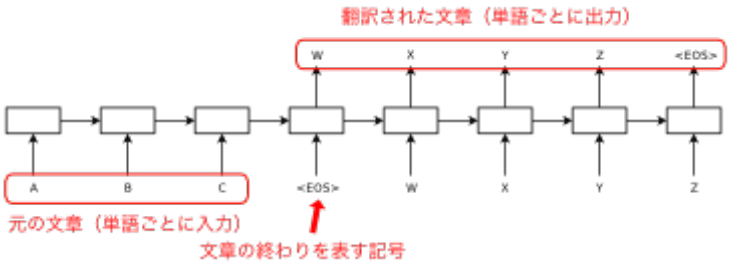


図 3.10 seq2seq の概略図　(Sutskever, Vinyals, & Le, 2014) より改変

する際にその重み付き和も使用します。このようにすると、ある出力単語を生成する際それに対応する入力単語に大きな重みを割り当てることで、その入力単語に「注意を向ける」ことが可能になります。この方法は (Luong, Pham, & Manning, 2015)[22] によって開発されたものです。

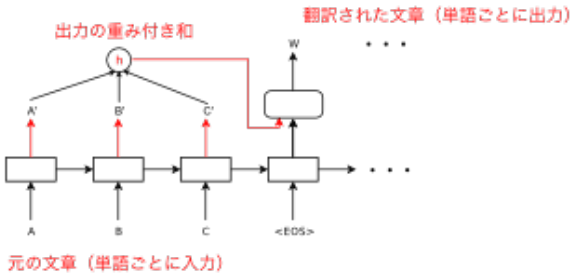


図 3.11 attention の概略図　(Sutskever, Vinyals, & Le, 2014) の素材を使用して作成

Attention は非常に効果的なので、手法は違えどそのアイデアは様々なニューラルネットワークで利用されています。もちろん CNN に適用したものも存在します [23]。

3. Virtual Brain Analysis

これまで紹介した CNN ですが、パラメータの数が多いこと・データから自動でパラメータが学習されることが原因で、CNN を設計している人たちにすら CNN がどのように情報を処理しているのか完全にはわかっていません。我々は CNN の構造と学習アルゴ

リズムを知っているものの、どのような「理論」によって画像から形や色、質感などを抽出し、画像内の物体の種類という抽象的な概念を認識しているかは明らかではないのです。それゆえ CNN は「Black box（中が見えない箱の意）」と呼ばれることもあります。

そこで、CNN がどのようにして物体認識を行っているかを明らかにするため脳科学から着想を得た方法を使って CNN を調べる手法が提案され、Virtual Brain Analysis[24] や Machine Psychology と呼ばれています。

Virtual Brain Analysis の一例として、(Simonyan, Vedaldi, & Zisserman, 2014) による activation maximization を紹介します。この章の最初に、Hubel & Wiesel の研究を紹介しました。彼らは視覚野の神経細胞の機能を知るため、ネコに様々な画像を見せ、神経細胞が物体の角度に反応していることを明らかにしました。これと全く同じように、CNN の一つ一つの神経細胞（に相当するもの、通常ユニットと呼ばれる）がどんな刺激に反応するかを調べることができないでしょうか？　もちろん可能ですが、CNN の場合はさらに洗練された方法で個々のユニットが最も反応する画像を生成することができます。詳細は省略しますが、次のような画像が得られます。画像は (Horikawa & Kamitani, 2017) から改変しました。

図 3.12 activation maximization によって作成された画像　(Horikawa & Kamitani, 2017) より改変

ここでは、CNN の各層に対して、その層の一つのユニットに対する activation maximization の結果を表示しています。初めの層（Layer 1）では単純な明暗の刺激に最もよく反応し、より高次の層になるほど複雑な刺激に反応することがわかります。Layer 8 で鳥のような物体が見えるのは、このユニットが鳥のカテゴリ判別に関わるユニットだからです。

このように脳を調べるのと似た手法を使うことで CNN がどのような情報処理をしているのかを調べることができます。現在、ニューラルネットワークはより複雑で大きなモデルへと進化しています。今後そのような複雑なモデルを解析・改善していかなければならなく立ったとき、Virtual Brain Analysis が大きな力を発揮するでしょう。

3.6 終わりに

以上でこの章の内容は全てです。この文章は 10 月 17 日に KaiRA の勉強会で発表した内容の一部を加筆・修正して作られました。今回の会誌では一般向けという性質上、数式や理論的な詳細については割愛している部分もありますが、KaiRA の勉強会ではより踏み込んだ内容を紹介しています。本節の内容に興味をお持ちの場合、ぜひ KaiRA の勉強会に参加してみてください。また、本書の内容について質問がある場合、kenmon112@gmail.com までご連絡ください。

第 4 章

ゲームと機械学習の関わりの事例

4.1 概要

4.1.1 初めに

「このゲームの AI が強すぎる」「この AI はアホだからこのステージは簡単にクリアできる」といったことを聞いたことはありますよね. ゲーム, 特にビデオゲームにおいては AI を活用している場合が大半です. というのも,「AI」という言葉は「知的な行動を人間のかわりに行う」という意味合いで広く扱われており, ゲームで使われる AI つまりゲーム AI はコンピュータゲームを作るにあたって基本となるからです. 皆さんがゲームにおいて「AI」という言葉を聞いて思い浮かべるのは, 格闘ゲームにおける戦闘を重ねるごとに強くなっていく NPC やオープンワールドゲームの NPC といったものだと思います. もちろんそれらは「ゲーム AI」です. しかし,「ゲーム AI」は実はそれだけではありません. 詳しくは次少節で説明しますが, ゲーム AI 自体には成長, つまり学習の要素自体はあってもなくても良いのですが, この章ではゲーム AI の中でも特に学習の要素を含んだものについての説明をさせていただきます.

また, ゲーム AI だけではなくよりゲームにおける音声合成といった少し面に近い領域で使われている事例の話もさせていただきます. ゲームハードの性能の上昇, 機械学習に関する新しい技術がどのようにしてゲームに関わってきたのか. 最近では VR ゲームも台頭してきて, この手の技術についても脚光を浴びています.

まとめると, 今回紹介する内容としては大きく 2 点に分けられます. 一つ目は機械学習を用いたゲーム AI について, 二つ目はシステム面に近い内容です. 具体例を多く取り扱い,「機械学習とか良く聞くけど実際にどうなん？」と思っている人に読みやすくなっていると思います. それに追従して, 技術というよりも知識という面に偏ってしまっているた

め, これで何かを学ぼうというよりも『読み物』 として楽しんで貰えたら幸いです.

4.1.2 3 種のゲーム AI

先ほど述べたように, 本題に入る前にゲーム AI について簡単に説明させていただきます. そもそも AI という言葉は「人間の代わりに作業をするもの」というように曖昧な表現で説明されます. これをゲームにおいて考えると,GM を用意したり対戦相手を用意していたアナログゲームからそれらの人から計算機に置き変えることでビデオゲームというものを生み出したのです.

さて, 実際に開発者たちがゲーム AI を実装しようとしたとき, 彼らはゲーム AI を体系化するべく次の 3 種類に分けました.

1. キャラクター AI
2. ナビゲーション AI
3. メタ AI

キャラクター AI とは, 環境から得られる情報を知覚部分を通じて, 知識を生み, 意思決定を行い, 行動を生成し, 実体を動かすことで環境に影響を及ぼす AI のことです. 要するに NPC のことを指します. キャラクター AI はエージェントと呼ばれることもあります. 格闘ゲームを例に挙げましょう. 皆さんがキャラクターを右に動かし距離を詰めたとします. するとエージェントはナビゲーション AI を通じて敵が距離を詰めてきたことを知ります. エージェントは敵が距離を詰めたことを記憶し, その行動に対して予めプログラムされた動きや計算して得られた結果に基づいて意思決定をおこないます. 今回は敵から距離をとるという意思決定をとるとします. エージェントが距離をとるという行動をなすには左に動くという行動をしなければいけません. それをエージェントに伝え, 左に動いてもらうというようになっています. 機械学習の一つである強化学習は主にここで用いられます.

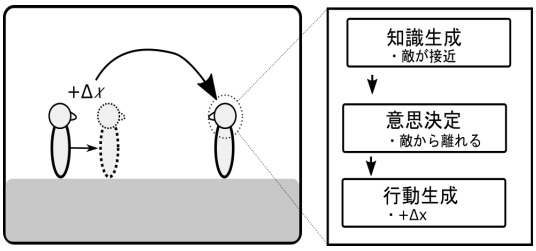


図 4.1 キャラクター AI の具体例

ナビゲーション AI とは, 環境から特徴的な部分を抽出し情報としてエージェントに伝える役目を持っています. ステルスゲームを例に上げましょう. ステルスゲームは文字通り敵から隠れながら目的を達成するゲームです. 図 4.2 の場合を考えてみましょう. 目が円いのがプレイヤーで, 目が楕円のものを敵とします. 敵は視界をもっていおり, 図 4.2 では斜線の領域にあたります. プレイヤーは左端から右端に動きたいのですが, 敵の視界には入ってはいけません. プレイヤーはもの陰に隠れながら移動する必要があるのです. この時にナビゲーション AI は隠れられる場所を図 4.2 の楕円で示している領域の情報を抽出します. これがナビゲーション AI の役割です.

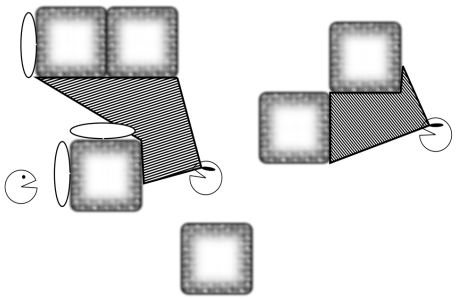


図 4.2 ナビゲーション AI 例 (ステルスゲーム)

メタ AI とはゲームの中で敵を生成するタイミングを調整するといった進行の管理を行う AI のことです. ゲームレベルデザインと関わってきます. 例えば, プレイヤーのプレイ情報から, よくゲームオーバーになるなら難易度を上げたり, 逆にどんどん難易度を上げていくのもこの AI が制御しているのです.

4.2　機械学習を用いたゲーム AI

ニューラルネットの発展などにより今後ゲームにもより機械学習が用いられる例が増えると個人的には考えています. しかし, 一般的に機械学習でゲーム AI を用いようとしても膨大な計算量や強化学習の不安定さが問題となっており, 実装している場合は現時点では多くはありません. その多くないながらもいくつかを紹介をさせていただきます.

Creatures

Creatures のシリーズ第一作目は 1996 年に Windows 向けに発売されました．このゲームはノルンと呼ばれる種族のエージェント達に言葉や行動を覚えさせてゆく育成ゲームです．学習器はキャラクター AI にて実装されています．エージェントには感覚運動調節と行動選択を出力する 2 種類のニューラルネットワークと生物の生化学をモデル化した「人工生化学」, そして神経活動と段階的な個体発生をモデル化した「ホルモンシステム」が実装されています．これらによってプレイヤーはノルン達を学習させ成長し, 新世代を生み出していくことができます．

知能を担うニューラルネットワークと脳のモデルの説明をしましょう．このゲームにおいては, ニューロンの集合の事を「ローブ」と呼んでいます．そしてローブとローブをニューロンによって接合することにより, 頭脳のニューラルネットワークを再現しているのです．図 4.3 を見るとノルンの脳のモデルが分かります．Attention ローブは注意ローブのことで, ノルンの視界に写る物体の中で注意を引かせることを可能にするローブです．Perception ローブは知覚ローブ,ConceptDecision ローブは意思決定ローブのことで, ノルンが知覚し, コンセプトローブを通じて意思決定を行います．学習においては遅延報酬型強化学習という手法が使われています．脳の興奮性シナプスと抑制性シナプスをそれぞれ報酬と罰としてその価値が今後最大となるように学習します．それぞれのシナプスはノルンの「欲求」の増減によって生成されます．例えば, ノルンが空腹になった場合を考えます．ノルンは「ものを食べたい」という欲求をもっています．しかしここで何も食べなければ, 時間とともに抑制性シナプスが分泌され, 罰が増えていきます．反対にものを食べると, 欲求が満たされ興奮シナプスが分泌され, 報酬を得ることができます．この報酬を増やすことを学習することで, 欲求を満たす行為を段々とするようになるのです．ものを覚えさせる時には,「痛みを減らす」という欲求が関与します．あるものに注意を向けさせた時, 新しいものならば単語を入力して, 覚えさせます．そしてもう一度見せて名前を言わせ, 間違っていれば間違っていれば痛み (マウスでクリック) を与えます．このようにして,

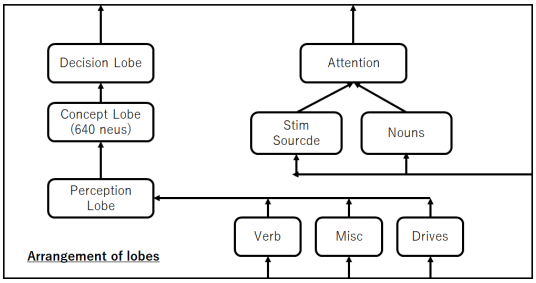


図 4.3 ノルンの脳モデル

痛みを与えられれば「痛みを減らす」という欲求が増加して, 抑制性シナプスが増え罰が増加します. このようにして言葉を覚えさせるのです.

SAMURAI SPIRITS

2019 年に PS4 で発売された対戦格闘ゲームです. ここでもキャラクター AI として実装されています. このゲームでは「ゴースト機能」とよばれる, プレイヤーから動きを学んだエージェントとプレイヤー自身が戦うことのできる機能があります. GAME CREATERS CONFERENCE'19 で開発者が語ったニューラルネットワークを簡単に説明します. 図 4.4 に示すようにエージェントに入力するデータは意思決定を定める 76 個の要素にして, 時系列データを扱うための Clockwork RNN と呼ばれるネットワーク, そして全結合層, ソフトマックス関数で 2 股に分け, ゲームコントローラーのようなボタン入力を出力させています. Clockwork RNN は LSTM よりも長い時系列を扱うことができるニューラルネットワークです. 初めは強化学習を用いたそうですが, 局所最適に陥っ

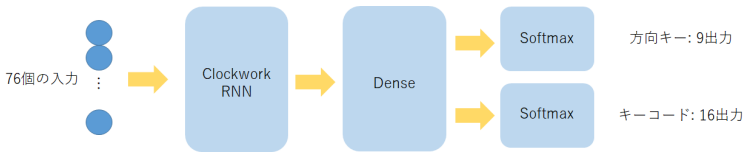


図 4.4 ゴーストの学習ニューラルネットワーク

てしまったそうです. だから手法を変えて, 時系列を考慮し, エージェントが画面の情報から入力を得て, コントローラーのキーを出力させることでこの問題はある程度緩和したそうです.

その他ビデオゲーム

BATTLEFIELD 1 ではエージェントの動きにおいて模倣学習が使われています. 模倣学習とは強化学習の一種で, エキスパートが手本を示しそれに追従するように学習させる手法のことです. ソーシャルゲームの逆転オセロニアでは, デッキアーキタイプの品質管理のためにトピックモデルと呼ばれる分析方法を使用しています. またグリムノーツでは, ゲームバランス調整のために遺伝的アルゴリズムを用いて, 指定されたバトルに対しパーティーの最適な編成を探していました.

近年ではソーシャルゲームやオンラインゲームが台頭して来ています. 過去には何かバグがあれば第 2 版やその時に売ってあるものを回収したりする必要がありました. しかし, 現在ではバグを直したものをそのままプラットホームを介して配布するといことができています. 機械学習の点から見ると, サーバ上で膨大な計算を要する学習を行い, その結

果をクライアントに送るという手法をとることで機械学習を有効に使っているのです.

4.3 技術として用いるゲーム

この節ではゲーム AI 以外で, 機械学習を用いている又は関係しているというような例を紹介させていただきます. しかし全てを紹介しようとしても仕切れないので, 今回は主に音声合成, 動作補間について説明します.

4.3.1 音声合成

ゲームにおいて音声合成技術自体は古くから使われています. 規則音声合成, 波形接続合成, 統計的パラメトリック音声合成の 3 つを紹介します.

規則音声合成は人間の声道をモデル化したしたような音声合成手法です. 例として, パルス波にフォルマント周波数だけを強調するフィルタをかけたりするものが挙げられます. この手法は昔のアーケードゲームのキャラクターのボイスだったり, 効果音に使われています.

波形接続合成は大量の音のデータを予め用意して, それを接続する手法です. ときめきメモリアルの Emotional Voice System(以下 EVS と呼ぶ) はこの技術を使っていると思われます. というのも,EVS が登場したときめきメモリアル 2 が発売された 1999 年はまだ統計的パラメトリック合成の手法は確率しておらず, また規則合成では表現できない自然さがあるからです. また大量の音声データを用いているという点も根拠になります.

統計的パラメトリック音声合成は音声データから生成モデルを作り, そのモデルをから音声を合成する手法です. 収録した音声データをもとに違う声質の音声も合成することができます. 近年, 機械学習を用いた音声合成がうたわれていますが, それらはこの方法に分類されます. CeVIO と呼ばれるプロジェクトの音声合成ソフトウェアにはこの技術が使われています.

めがみめぐり

このゲームは日本の駅を双六盤として, エージェントである「つくも」と旅をしながら彼女に言葉を教えてあげて成長させていく内容になっています. 2016 年に 3DS のゲームソフトとして発売されました. これはあくまで私見ですが, この「つくも」の音声合成には統計的パラメトリック合成が使われていると考えられます. その理由をいくつか説明させていただきます. 一つ目に容量が少ないということです. ハードが 3DS ですから, みなさんも容易に想像が着くと思います. 合成用の生成モデルさえ作ってしまえば, 波形接続型のものより容量は少なくてすみます. 二つ目に, 収録時間が少ないという点です. このめがみめぐりの音声合成エンジン,「めがみスピークエンジン」には東芝の「RECAIUS 音

声合成ミドルウェア ToSpeak^{TM}」というミドルウェアが組み込まれています．この製品のホームページを見てみると, 特徴の一つとして短時間の収録で合成音を生成できるとのことです．前小節の通り波形接続型音声合成のためのデータは大量に必要になります．実際にときめきメモリアル 2 の EVS の製作のために凄い時間がかかったと聞きます．この 2 点から, めがみスピークエンジンには統計的パラメトリック合成が使われていると考えられます．このエンジンの特徴の一つとして感嘆詩や吐息を再現しているということがあげられます．より人間的であることを追求しているのが分かります．

4.3.2　動作補間

近年は VR ゲームもどんどんと増えてきています．それと同時に VR ゴーグルにおいて格安からお高いものまでと出てきています．その差はゴーグルやコントローラーの自由度の差であったり, フレームレートの差であったりします．そのため VR 空間内での動作のなめらかさ等に違いが現れたりしてきます．他にもキャラクターの動作処理おいて, 人が全てをリギングしていてはとても時間がかかります．そのため機械学習によりそれらの効率を上げようという取り組みもあります．Computer Entertainment Developers Conference 2019(以下,CEDEC2019 と呼ぶ) で登壇された株式会社 CRI・ミドルウェアがお話になったのを例にあげましょう．彼らは音声をもとにキャラクターの口の動きを自動生成するツールを作ろうとしたそうで, 線形予測とニューラルネットワークを用いた例として 2 種類紹介していました．

他にも色々とありますが, 一つだけ実例を紹介します．

CYBER TENNIS

このゲームは VR ゴーグル,Oculus Go でプレイできる VR テニスです．Oculus Go 付属のコントローラーを振ることにより, 実際のテニスラケットを振るようにプレイを楽しめるというものです．しかし,Oculas Go のコントローラーは自由度 3 となっていて, 入力情報が少なく誤差で生じやすくなっています．そのため, このゲームでは 3 層の単純なニューラルネットワークを用いて問題を解決しました．この 3 つの自由度, 水平角, 天頂角,xyz 座標の加速度を入力値として 5 種類の打ち方を出力するネットワーク用いているようです．

4.3.3　今後の動向

今後の注目されている点として, 自然言語処理, 検出の二点を紹介します．まずは自然言語処理についてです．確認ですが, 自然言語というのは我々が日常で話している言葉のことです．つまり自然言語をゲームに用いることでより「没入感」や「共感」を味わえるの

です．ゲームの NPC は自然言語で語りかけてきますよね? それはやはり共感といった感情を抱いて欲しいからです．しかし RPG などのプレイヤーは予め決められた選択肢を選ぶしかありません．そうなると, 自然言語処理処理, つまりプレイヤーの言葉を解析する技術を用いたゲームというのはよりその感情を味わうことができます．1999 年に PS2 で発売された音声対話ゲーム, シーマンは自然言語処理を使っています．主に使用しているのはシーマンの方ですが, 言葉の合間やイントネーションの違いにより発話者の意図を変えることができるように, それを認識したり発声させたりしています．最も, 今のところ計算機は文の意味を理解している訳ではありませんが.

次に検出についてです．今回の検出においては主に顔検出について説明します．カメラ等に搭載されている事実から顔検出は昔からあることだとは分かると思います．機械学習を用いなくとも顔検出はできますし, もちろん機械学習を用いても顔は認識出来ます．では機械学習を用いることの利点は何かというと, 顔の特徴を得ることができるという点です．CEDEC2016 で登壇された NTT PC は人の表情を読む機械学習についてデモとお話になられていていました.

4.4 まとめ

ゲームにおける機械学習の事例を大きく 2 種類に分けて見てきました．ゲーム AI やテスター AI といった分野とユーザーに近い分野の 2 種類です．前者の分野では現在ではキャラクター AI の意思決定に多く機械学習やアルゴリズムが考えられてきています．近年はメタ AI への活用も試みられています．後者の分野では様々なゲームシステムを作成するために研究がなされてきました．プレイヤーを楽しませる AI だけではなく, 開発者の手助けとなるものもこの分野で研究されています.

さて, この章を通じて機械学習やゲーム AI について興味を持っていただく, または事例を理解していただいたら幸いです．もっと活用事例を知りたいという方には CEDEC の資料や GDC のレポートを探していただいたら良いと思います．そして, 申し訳ないことに私は全く原理的なことを話しておりません．もし, 強化学習とは一体どのような学習なのか,LSTM や RNN とは一体何なのかといったことを知りたい方は自身で調べていただいたら幸いです．KaiRA2019 年後期はゼロから作る Deep Learning 2 の輪読会をやっていて内容は自然言語処理 (RNN や LSTM 含む) となっております．気が向いたら, ホームページから参加方法の手順で参加していただいたら嬉しいです.

最後におまけとして, 強化学習で大事な考えとなるマルコフ決定過程を主にお話しします.

4.5　おまけ: 強化学習の基礎

2 節で度々出てきた強化学習の基礎となるマルコフ決定過程と強化学習が学習するものについて説明します. そもそも, 強化学習とは機械学習の学習法の一種です. 一定のルールに従う環境において, エージェントが行動を起こし, その状態と行動に応じた報酬や罰を得て, 報酬を最大化するように報酬の計算法や報酬の得方, つまりエージェントの行動方針を学習する方法です. そのルールというのは, 次状態への遷移が現状態と行動のみに依存し, 報酬は現状態と次状態に依存するというものです.

この一定のルールのことをマルコフ性といい, この環境をマルコフ決定過程 (以下,MDP と呼ぶ) といいます. MDP には 4 つの要素があります. 一つ目は状態 s です. エージェントは環境内のある状態として必ず存在します. 二つ目に, 行動 a です. エージェントは定められた行動のみをとります. 三つ目に, 遷移関数 T です. 現状態と行動を入力に, 次状態の遷移確率を出力する関数です. 四つ目に, 報酬関数 R です. 現状態と次状態を入力に, 報酬を出力します. 行動を引数にとることもあります.

さて, 強化学習で行うことはこの報酬の合計を最大化することです. 状態を遷移するごとに最大の報酬を得ることができれば自ずと合計の報酬は最大になるとは限りません. 目の前の利益をとるか, 目先の利益をとるかといったことが起こるからです. このことから, エージェントは未来の報酬を考慮する, つまり未来の報酬を見積もって計算する必要があります. 数式で説明しましょう. 時刻 t において, 報酬の合計を G_t, 即時報酬を r_t, 割引率をγ とすると,

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{T-t-1} r_T \tag{4.1}$$

のように表されます. 未来の時刻での報酬はあくまで見積もったもので割引率を用いてその大きさを小さくしているのです. 不確定要素の大きい遠い未来になればなるほど割引率は大きくなり, 計算する合計報酬に与える影響を小さくしています. 式 4.1 を整理して書くと,

$$G_t = r_{t+1} + \gamma G_{t+1} \tag{4.2}$$

となります. これにより G_{t+1} の値の算出を一時的保留に出来ます.G_t は時刻 t での価値と呼び, この価値を計算することを価値評価と言います. しかし, この式だけでは問題があります. それは前の式だと r_{t+1} の値が必ず一意に定まる必要があるという点です. それを解決するために計算する際は r_{t+1} の期待値をとるということです. 本当はいくつか行動の選択肢が存在して, そのうち状況に応じた行動が最も選択肢の中で確率が高い, と考えたほうが自然ですよね. つまりとる行動は確率によって決まると考えます. この行動の確率分布のことを方針 π と呼びます. 方針 π を用いて時刻 t における合計の報酬 $V_\pi(s_t)$ を求めてみましょう.

$$V_\pi(s_t) = E_\pi[r_{t+1} + \gamma V_\pi(s_{t+1})] \tag{4.3}$$

となります. 先ほど述べたように行動は全て確率で選ばれると考えるため期待値をとっています. 行動確率 $\Pi(a|s)$, 報酬関数 $R(s, \acute{s})$ をもちいると前の式を一般化すると,

$$V_\pi(s) = \sum_a \Pi(a|s) \sum_{\acute{s}} T(\acute{s}|s, a)[R(s, \acute{s}) + \gamma V_\pi(\acute{s})] \tag{4.4}$$

のように書き換えることが出来ます. この方程式のことを Bellman 方程式と呼びます. この式から強化学習で更新していく式を導くことが出来ます.

第 5 章

統計力学から見る活性化関数と誤差関数

5.1　序論

5.1.1　はじめに

本稿は今年 6 月に出た『ディープラーニングと物理学』の「3.1　誤差関数とその統計力学的理解」の解説である。その内容は統計力学の形式から深層学習で一般的に使われている誤差関数と活性化関数を導出するというものである。例えば、誤差関数であれば二乗和誤差、クロスエントロピー、活性化関数であればシグモイド関数、ソフトマックス関数といったものがあるが、それらをある物理系での統計力学な振る舞いから導出することができる。従来の深層学習の本では誤差関数、活性化関数の具体形は天下り的に導入されることが多い。誤差関数が正解と機械学習の状態との「距離」として機能すべきこと、活性化関数の非線形性により表現能力が上がることが強調されることが多いが、その条件を満たすだけならいくらでもほかの関数の候補が存在する。それに対し本稿での統計力学を用いて導出する手法では、simple な物理系を設定するだけであとは一意的に形が決まる。

『ディープラーニングと物理学』は様々な深層学習での概念を天下り的に導入するのではなく、導出することに重きが置かれており納得していきながら読み進めていくことができる本である。是非たくさんの人に読んでもらいたい本ではあるのだが、物理の人間向けに書かれている部分があり、そこの癖がちょっと強い。物理の内容に絡めて説明されることも多いので、物理以外の人が読むと少し戸惑うかもしれない。本稿の目的はそのギャップを埋めることである。そのためにはじめに統計力学について説明をする。人工知能研究会の会誌なのに何で統計力学やんなきゃいけないだよと思われる方もいるとは思うが、統計力学の基礎を学ぶことは決して損にはならない。というのも統計力学は多数の粒子の系

からエネルギーなどの欲しい情報を取得することができるフレームワークであり、それは情報理論とも関係が深い。本稿は物理になじみのない方を対象に書かれたものである。物理に慣れている方は 5.2 を飛ばして読むことを勧めるが、一番のおすすめは直接本を購読することである。

5.1.2 学習

機械学習において「学習」とはどのように定義されるか？それは大まかに言えばあるデータ分布を再現するような分布を人の手をほとんど借りることなく自律的に構成することである。例えば、掛け算の学習する場合を考えてみる。学習の仕方として一番単純なのは丸暗記をしてしまうことである。実際、九九はそういった戦略に基づいており、正にデータ分布を自分の頭の中に構成することといえる。

しかしこれはデータの数が高々 8 1 個しかないからできることであって、「11112 × 49435」といった計算まで暗記することはできない。正確な値が知りたければ紙と鉛筆で計算することになるが、今回は暗算しかできない場合を考える（紙と鉛筆で計算することを言い換えれば経験から帰納的に演算についての法則を導き、それに基づいた手法で細かいタスクに分割して処理するということにあたるが今回はそのような複雑な例は考えない)。暗算だと厳密な値を求めるのは厳しいが、全く見当がつかないかというとそうではない。例えば経験的に掛け算の一方の数字が偶数であると答えも偶数であることを知っていれば、「11112」が偶数であることから、値が偶数であると予想できる。さらにもっと言えば「49435」は一の桁が 5 であることから答えが 5 の倍数であること、「11112」の全部の桁の数を足すと 1+1+1+1+2=6 と 3 の倍数になることから答えが 3 の倍数であること、 5 桁と 5 桁の掛け算だから答え 9 ～ 11 桁の数になることなど様々な予想を立てることができるのである。

こんな風に全く分からないわけでなくぼんやりと答えの予想を立てることができるので、これはぼんやりとした分布を自分で構成したことになる。機械学習の場合、ぼんやりとは決定的ではなく確率的であるということを指す。つまり「学習」とは入力「11112 × 49435」に対して答えは何々であるという確率を求めることと言い換えることができる。

5.1.3 統計力学の使い方

次に問題となるのがそのような確率分布を構成する方法である。そこで今回はある物理系を考えることでそのような確率を求めたいと思う。今の方針を聞いて、物理と確率という組み合わせに違和感を持たれる方もいるかもしれない。惑星の軌道は運動方程式から決まるし、時計の振り子はいつも同じように振れるのであるのだから、物理現象はすべて決

定的であり確率という概念が入ってくる余地がないのではないかと。しかし、仮にそうであったとして決定的に物理系を解析する必要があるかというと必ずしもそうではない。例えば、標高が高い場所は空気が薄いことを説明する時を考える。決定的な物理理論のみで説明しようとするとき、全粒子の運動についての方程式を解く必要があるが実際これは困難である。そもそも運動方程式での答えを知るには現在の粒子の状態を予め知っている必要があるので現実的ではない。それに対して、空気が薄いのは粒子が重力に引っ張られて標高が高い位置にいる確率が低いからであると言えばこれは簡単に説明できる。[*1] 勿論、都合のいい確率をでっちあげるのではなくて、「等重率の原理」に従って厳密に求められる。これが統計力学という物理の一分野のでの考え方である。詳しくは後の章で説明する。

ここで重要なのは粒子の状態を完全に決めなくても、空気の薄さといった物理量はほぼ正確に予測できるということである。これは先程の学習の話でいえば「11112 × 49435」の正確な答えはわからないが大体 9 11 けたの数であると予測できるのと近い。

今回の方針は次のとおりである。ある物理系を用意し、それに何かしらの操作を行う。その操作が機械学習でいえば入力、先程の例で言えば「11112 × 49435」に対応している。すると、物理系はそれに応じてある状態になり、我々は物理量を測定する。ちょうどいい物理系を用意してくれば、測定した物理量は高い確率で「9 ～ 11 桁の数」となるだろう。その平均値を出力とし、理想的には「549321720」となってくれればいいことになる。このような方針で進めていったとき深層学習では使われることが多い活性化関数や誤差関数が自然と出てくることを後半で確かめる。

5.2　統計力学の基本的な説明

ここでは統計力学及び解析力学の基本的な事項を説明する。

5.2.1　解析力学

先ずは物理での基本的なフレームワークについて説明する。このフレームワークを数理的に扱う分野を**解析力学**という。ある物理系を考えるとき先ず**ラグランジアン**

[*1] ここで注意すべきなのは統計力学で確率が導入されるのは系の状態を完全に把握できないゆえに導入されるものであって、量子力学とは直接は関係がない。ここでいう状態は古典力学では位置や速さなどで指定されるものである。それに対し量子力学の「状態」とは「波動関数」で指定されるものである。「波動関数」とはある「測定」をしたときにどんな「測定値」をどのくらいの確率で返すかという情報を持ったものであり、適切な数学的な処理をすることで理論的にそれは求まる。量子力学は波動関数の性質から確率的な側面が見受けられるが、波動関数という状態自体はシュレディンガー方程式という運動方程式で決定論的に記述されるのである。

$L(q(t), \dot{q}(t))$ というものを定義する。但し、$q(t)$ は粒子の位置、$\dot{q}(t)$ は粒子の速度を表す。ラグランジアンは今取り扱っている物理系がどのようなものであるかを数理的に表すモデルである。そしてこのラグランジアンから粒子の運動の経路が、**最小作用の原理**によって決められる。

最小作用の原理

作用 $S[q(t)]$ を次のように定義する。

$$S[q(t)] = \int_{t=t_i}^{t=t_f} dt L(q(t), \dot{q}(t)) \tag{5.1}$$

粒子の経路は作用が最小（極小）になるような経路を選ぶ。但し、固定条件として時刻 $t = t_i$ と $t = t_f$ での状態は設定されているとする。解析力学ではこれを原理として導入し、最小作用の原理という。

直感的には最小作用の原理のようなものを導入するモチベーションが分からないと思う。しかし、物理現象を数理的な解析する上ではシンプルな形を与えてくれる。例えば、屈折率が連続的に変化するような媒質中を進む光の経路は最短時間（正確には極小時間）で移動できる経路を進んでいくことがわかっている。

そしてラグラジアン $L(q(t), \dot{q}(t))$ から**ハミルトニアン** $H(q(t), p(t))$ というものが導入される。

ハミルトニアン

正準運動量 $p = \frac{\partial L}{\partial \dot{q}}$ としたときハミルトニアン H は次のように定義される

$$H(q(t), p(t)) = \dot{q}p - L \tag{5.2}$$

ハミルトニアンはラグランジアンを q, p についてルジャンドル変換したもの

ハミルトニアンが便利なのは、一つはハミルトニアンが粒子のエネルギーという物理量を表す点。もう一つはハミルトニアンを使うことで粒子の運動が連立微分方程式の形で表すことができるようになる点である。この微分方程式を**正準方程式**という。

正準方程式

粒子の運動は次の正準方程式に従う

$$\frac{dq}{dt} = \frac{\partial H}{\partial p} \tag{5.3}$$

$$\frac{dp}{dt} = -\frac{\partial H}{\partial q} \tag{5.4}$$

例としてばね振動の場合を考える。但し、ラグランジアンから始めるのは面倒くさいの

でハミルトニアンの形は与えられる元として議論する。振動していない時のばねの釣り合いの位置を 0 とすると

$$H = \frac{p^2}{2m} + \frac{1}{2}kq^2 \tag{5.5}$$

正準方程式を解くと

$$q = A\sin t\sqrt{\frac{k}{m}} \tag{5.6}$$

$$p = A_0\sqrt{mk}\cos t\sqrt{\frac{k}{m}} \tag{5.7}$$

実際にばねが振動している様子が、表されていることがわかる。

以上が解析力学の基本的な内容であるが、なぜこのような手法をとるのかを簡単に説明する。一つは考えている物理系の対称性から保存量が得られるからである。

- 時間並進対称性 $\rightarrow$ エネルギー保存
- 空間並進対称性 $\rightarrow$ 運動量保存
- 回転対称性 $\rightarrow$ 角運動量保存

逆に対称性を課すことでラグランジアンの形が制限される点も利点である。例えば、時間並進対称性、つまり物理法則が時間によらないということを要請すれば、ラグランジアンは直接時間に依存しない関数にしなければならない。このような利便性から物理学の多くの分野で受け入れられているのである。

5.2.2　統計力学の基礎

次に箱の中の気体の圧力やエネルギーを求めることを考える。原理的には上で導入した解析力学の手法を用いて 10^{23} 個の粒子の運動を考えてやればいいことになるがそれは現実的ではない。序論でも述べたが粒子の運動を記述するためには先ず現在の粒子の状態を調べる必要があるからである。加えて、数学的に三体以上の力学系の運動はカオス的、つまり極めて鋭敏な初期値依存性を持つことが知られている。

しかし、人類が気体の物理的なふるまいについて何も予言することができないかというとそうではない。実際、蒸気機関でたくさんのモノ・人を運んだり、エアコンで部屋の温度調節ができるのは気体の物理的振る舞いをきちんと把握しているからである。

蒸気機関の発達とともに発展していった物理の分野に**熱力学**というものがある。熱力学は平衡状態であれば多数の粒子からなる気体について圧力や温度といったマクロな物理量だけで様々な解析をしていくことができる理論体系である。ここでいう**平衡状態**とは時間や場所にマクロな物理量が依存しないという意味である。例えば、箱の中の気体の圧力は場所や時間によらず同じである。[*2]

[*2] 厳密には長さスケールと時間スケールが設定されてから平衡状態が定義される。例えば、遠い遠い銀河か

しかし、このマクロな物理を扱う熱力学はミクロな粒子の物理法則から得られるものであるべきである。そこでミクロな粒子の状態にある確率分布を仮定し、そこから導出されるマクロな物理量を調べることで、ミクロな物理とマクロな物理をつなげるのが統計力学という学問である。

さて、ミクロな物理の状態に確率分布を導入したいが決定論的な解析力学の手法では確率を導入することができない。そこでまず等重率の原理という仮定を認めることで確率を導入していく。

等重率の原理

N 個の粒子の状態は $q_1,\cdots,q_N,p_1,\cdots,p_N$ で指定されるとする。この時、状態 $q_1,\cdots,q_N,p_1,\cdots,p_N$ と状態 $q'_1,\cdots,q'_N,p'_1,\cdots,p'_N$ のエネルギーが等しいならば二つの状態をとりうる確率は等しい。これを等重率の原理という。

$q_1,\cdots,q_N,p_1,\cdots,p_N$ が作る $2N$ 次元の数学的な空間を相空間という。相空間の概念を用いれば等重率の原理は数理的に言い換えることができる。

等重率の原理

$q_1,\cdots,q_N,p_1,\cdots,p_N$ の周りの体積 $dq_1\cdots dq_N dp_1\cdots dp_N$ の中に状態がある確率は次のように表される

$$P(H(q_1,\cdots,q_N,p_1,\cdots,p_N))dq_1\cdots dq_N dp_1\cdots dp_N \tag{5.8}$$

5.2.3 ボルツマン分布

次に平衡状態にある物理系を考えることで等重率の原理からボルツマン分布を導出する。そのためにまず N 個の粒子が入った箱を M 個用意して、それを積み並べることで大きな箱を作る。但し、壁を挟んで箱同士の間で弱い相互作用をするため、エネルギーのやり取りをすることができる。この時、M 個の箱を積み重ねて作った大きな箱は MN 個の粒子からなる系と考えることができ、この大きな箱が平衡状態になったとする。以下、この大きな箱のことを大箱という。

さて、一つの箱に注目したとき状態は 2N 次元の連続的な相空間の一点として表される。ただこのままだと便宜上扱いにくいので、相空間を体積 h^N の空間で区切り、状態を離散的にする。そしてその状態に番号を付けていく。状態 i の時のエネルギーを e_i とする。

次に大箱について考える。状態 1 である箱が M 個ある箱のうち m_1 個、状態 2 である箱が m_2 個、$\cdots$ という状況を考えよう（図**??**）。この時の場合の数はいくつあるだろう

ら我々の住む地球を含む星の集団の動きを録画し、早回しをすれば気体分子のようにふるまうであろう。この時、星の集団は平衡状態とみなされる

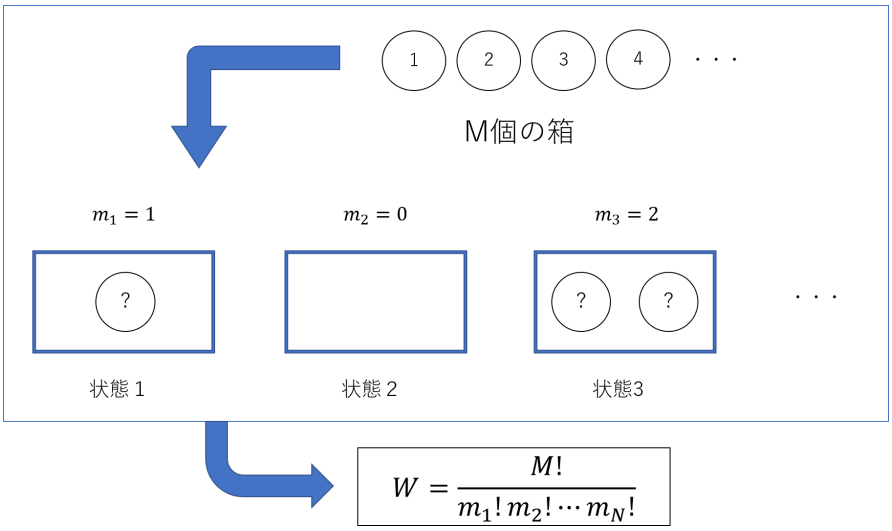


図 5.1 状態数 W についての考察

か。例えば $m_1 = 2$ の時、状態 1 である箱が箱 1 と箱 2 の時や箱 12 と箱 32 の時など様々な場合が考えられる。この場合の数を状態数 W とする。これは簡単な考察で次のようになることがわかる。

$$W = \frac{M!}{m_1! m_2! \cdots m_s!} \tag{5.9}$$

このとき、平衡状態において状態数が最大となることを示すことができる。ここで注目すべきなのは平衡状態においてマクロな物理量 O は時間や場所に依存しないという点である。

まず、場所に依存しないということはある特定の箱がある特定の状態であるということに O は直接依存しないということである。仮に O は箱 1 の状態に依存していたとする。この時大箱を縦半分に大箱 A と大箱 B に分けたとしよう。大箱 A には箱 1 が含まれるとき、勿論大箱 B には箱 1 は含まれない。この違いと O は箱 1 の状態に依存しているということから大箱 A、B のマクロな物理量について $O_A \neq O_B$ という関係が得られるはずだがこれは「マクロな物理量が場所に依存しない」という平衡状態の性質と矛盾する。つまり O はミクロな状態に直接依存するのではなく、ミクロな状態の分布に依存するということである。つまり

$$O = O(m_1, m_2, m_3, \cdots) \tag{5.10}$$

そしてこのことから、粒子は刻一刻と状態を変化させているのになぜ O はいつも一緒である理由を説明することができる。それは分布 $m_1, m_2, m_3, \cdots$ がいつも一緒だからである。なぜ一緒なのか、平衡状態の時の分布 $m_1, m_2, m_3, \cdots$ の状態数がほかの分布の状態数に比べて格段に大きいからである。「はずれ」ばっかのくじを引いたら大体いつも「は

ずれ」なのと一緒である。これは等重率の原理によってすべての状態が同じ確率を持っていることができることからより厳密に正当化される（大箱の全エネルギーが常に一定であることは解析力学により要請されることに注意）。平衡状態の状態数は M が大きくなるにつれてより高いピークとなることが知られている。

さて、平衡状態の時、状態数が最大となることを手掛かりに平衡状態の分布を導出する。そのために近似を用いて数学的な処理をしやすくする。

$$\begin{aligned}\log W &= \log M! - \sum_i \log m_i! \\ &\simeq M \log M - \sum_i m_i \log m_i \end{aligned} \tag{5.11}$$

但し、次の近似式を用いた

スターリングの公式

$$\log N! \simeq N \log N - N \qquad (N \gg 1) \tag{5.12}$$

次に状態数 W を最大にする m_i を求めるが、次の二つの量が固定されていることに注意する

$$E = \sum_i m_i \epsilon_i \tag{5.13}$$

$$M = \sum_i m_i \tag{5.14}$$

以上の点に注意して、ラグランジュの未定乗数法により求める。つまり次の量が最大となる m_i を求めればよい。

$$\log W + \alpha(M - \sum_i m_i) + \beta(E - \sum_i \epsilon_i m_i) \tag{5.15}$$

得られる結果は次のとおり。

$$m_i = e^{-\beta\epsilon_i - \alpha - 1} = \frac{M}{Z} e^{-\beta\epsilon_i} \tag{5.16}$$

この時ある一つの箱に注目したときに状態 i である確率が $\frac{e^{-\beta\epsilon_i}}{Z}$ であることがわかる。M の固定条件より

$$Z = \sum_i e^{-\beta\epsilon_i} \tag{5.17}$$

こうして得られる分布がボルツマン分布である。

ボルツマン分布

$$\frac{e^{-\beta\epsilon_i}}{\sum_j e^{-\beta\epsilon_j}} \tag{5.18}$$

ボルツマン分布は一つの箱に注意したときの確率分布である。これがたくさん集まると

き、中心極限定理よりマクロな物理量の揺らぎはほぼゼロになり熱力学的な現象が説明できる。パラメータ β についてはボルツマンの公式、

$$S = k_B \log W \tag{5.19}$$

と熱力学の公式

$$\frac{\partial S}{\partial E} = \frac{1}{T} \tag{5.20}$$

によって

$$\beta = \frac{1}{k_B T} \tag{5.21}$$

であることが示される。この関係を示すのには熱力学的な議論が必要になるため具体的な証明は省略する。その代わりに β $\beta(T)$ であることは次のような議論で示唆される。M_1 個の箱から成る系と M_2 個の箱から成る系の間に弱い相互作用があるとする。この時

$$W = \frac{M_1!}{m_1! \cdots m_s!} \cdot \frac{M_2!}{m'_1! \cdots m'_{s'}!} \tag{5.22}$$

$$\log W = M_1 \log M_1 + M_2 \log M_2 - \sum_i m_i \log m_i - \sum_i m'_i \log m'_i \tag{5.23}$$

但し次の量が固定される

$$M_1 = \sum_i m_i, \qquad M_2 = \sum_i m'_i \tag{5.24}$$

$$E = \sum_i \epsilon_i m_i + \sum_i \epsilon'_i m'_i \tag{5.25}$$

ボルツマン分布を導出したときのように、ラグランジュの未定乗数法によって次の量が極大となる場合を調べればよい。

$$\log W + \alpha_1 (M_1 - \sum_i m_i) + \alpha_2 (M_2 - \sum_i m'_i) + \beta (E - \sum_i \epsilon_i m_i + \sum_i \epsilon'_i m'_i) \tag{5.26}$$

これより得られる結果は以下の通り

$$m_i = \frac{M_1}{Z_1} e^{-\beta \epsilon_i}, \qquad m'_i = \frac{M_2}{Z_2} e^{-\beta \epsilon'_i} \tag{5.27}$$

これより二つの系でパラメータ β を共有していることがわかる。このとき、二つの系の間に相互作用があると平衡状態の時、温度が等しくなることに注目する。実際、温水と氷が入った袋を十分長い間放置すると次第にエネルギーが移り同じ温度になる様子は日常生活でもなじみの現象であろう。温度が二つの系で等しくなることから共通のパラメータである β が温度の関数であることが予想される。

ここでボルツマン分布の基本的な物理的性質を見る。すべての状態の中で最も低いエネルギーを ϵ_0 とする。この時、

$$\begin{aligned} P(\epsilon_i) &= \frac{e^{-\beta\epsilon_i}}{\sum_j e^{-\beta\epsilon_j}} \\ &= \frac{1}{\sum_j e^{-\beta(\epsilon_j-\epsilon_i)}} \\ &= \frac{1}{e^{-\beta(\epsilon_0-\epsilon_i)} + \sum_{j\neq 0} e^{-\beta(\epsilon_j-\epsilon_i)}} \end{aligned} \tag{5.28}$$

これより低温極限、つまり $T \to 0$、$\beta \to \infty$ の時を考えると

$$\begin{cases} P(\epsilon_i) \to 0 & (\epsilon_i \neq \epsilon_0) \\ P(\epsilon_i) \to 1 & (\epsilon_i = \epsilon_0) \end{cases} \tag{5.29}$$

逆に高温極限、つまり $T \to \infty$、$\beta \to \infty$ の時を考えると

$$P(\epsilon_i) \simeq \frac{1}{\sum_j e^{-\beta\epsilon_j}} \tag{5.30}$$

つまり、確率分布がエネルギーに依存せず一様となる。また二つのエネルギー状態があり $\epsilon_i > \epsilon_j$ の時、

$$\frac{P(\epsilon_i)}{P(\epsilon_j)} = e^{-\beta(\epsilon_i-\epsilon_j)} \leq 1 \tag{5.31}$$

以上の考察をまとめると、低温の時は低エネルギー状態が支配的となるが高温になるにつれ高エネルギー状態も現れ、十分高温になればエネルギーによらずすべての状態がほぼ同じ確率で現れ始める。

次に情報理論的側面からボルツマン分布導出の過程を振り返り、これがどのような分布であるのかを把握する。物理系としては M 個の箱が各々自由に状態を変化している系を考えた。よって大箱全体も状態が変化している訳だが、分布 m_i に注目し状態数 W が最大となるような大箱の状態を平衡状態とした。W が最大、つまり「ありきたり」で「当たり前」な m_i であるということである。情報理論では「ありきたり」で「当たり前」な事象に対して情報量が少ないという。情報量が最も小さくなるような分布を「無情報分布」という。但し、拘束条件が全くない場合一様分布となり意味が無い。それに対し、今回は系全体のエネルギー保存と箱の個数 M の保存が要請されることによって意味のある分布となる。まず箱の個数の保存の要請によって確率が全状態について足し上げると 1 になる。これは本質的ではない。重要なのはエネルギー保存という要請がされることで、これによってパラメータ β が導入され、指数関数的なボルツマン分布の形になるのである。

以上がボルツマン分布の導出、及び解説である。次にこれを用いて深層学習で用いられる活性化関数、損失関数を導出していきたい。

本筋とは脱線するが、ここで次のようなボルツマン分布の応用例を考える。クリスマスのプレゼント交換会は次のようなルールに従っている。「交換会」とは言っても、参加者に認められているのは「交換」ではなく一方的に「授与」することだけ。会の間は自由に「授与」することができ、誰かに「授与」されたものを他の誰かに「授与」しても構わないとする。十分時間がたった時の各人の持っているプレゼントの個数についての分布を考える。先程の物理系の例を考えると「授与」が箱から箱にエネルギーが移るのに対応しており、会場全体にあるプレゼントの個数が時間がたっても変化しないのが系全体のエネルギー保存に対応する。従ってプレゼント個数の分布はボルツマン分布になる。ボルツマン分布の性質から多くの人は 0、ごく一部の人が多くのプレゼントを持っているような状況になることがわかる。因みに交換会が始まったときに一人一つずつプレゼントを持っていたとすると $\beta = \log 2$ と求めることができ、プレゼントを n 個持っている人は全体の 2^{n+1} 分の 1 となる。現実には人には意思があるのでこのようにはならないと思われる（プレゼントが 0 はつらいので何とか避けようとするはず)。

5.3　活性化関数と誤差関数

5.3.1　基本的な戦略

ある物理系を用意するとハミルトニアンから統計力学のフォーマリズムに従って確率分布を導入することができる。深層学習ではデータ分布に近い確率分布を構成するのが目的である。基本的な戦略は次のようになる。

1. 入力 $\vec{x}$ を外場[*3]、出力 $\vec{d}$ を物理的自由度としたときのパラメータ付きハミルトニアンを用意する
2. そのハミルトニアンでのボルツマン分布を求める
3. そのボルツマン分布での出力の平均値 $<\vec{d}>$ を求めることで活性化関数が導出できる

[*3] 外場は抽象的な言葉として用いているので必ずしも電場や磁場のようなものと結びついてないことに注意。今回はあくまでこれは物理系に作用するもの、物理系の振る舞いを変化させるものという意味で用いている。

4. ボルツマン分布とデータ分布とのカルバック・ライブラー距離を求めることで損失関数が導出できる

以上の手順で活性化関数と損失関数を導入する。

5.3.2 二値出力

まず二値出力の場合、つまり入力が $\vec{x}$、出力が $d = 0 or 1$ となる場合を考える。例えば、$\vec{x}$ が画像、画像が猫の画像であれば $d = 1$、それ以外の時は $d = 0$ となるような場合がこれにあたる。この時ハミルトニアンとして次のようなものを考える。

ハミルトニアン

$$H = -(\vec{J} \cdot \vec{x} + J)d \tag{5.32}$$

この形の相互作用は実際様々な物理現象でみられる形である。例えば、二つの粒子がそれぞればねに独立につながれているときはハミルトニアンのポテンシャル部分は

$$H = \frac{1}{2}k{x_1}^2 + \frac{1}{2}k{x_2}^2 \tag{5.33}$$

となるが粒子同士でばねがつながっている、つまり粒子間で相互作用があるときは

$$H = \frac{1}{2}k(x_1 - x_2)^2 = \frac{1}{2}k{x_1}^2 + \frac{1}{2}k{x_2}^2 - kx_1x_2 \tag{5.34}$$

となる。このことから kx_1x_2 は相互作用を表し、k はその強さを示す。式（5.32）はこれに対応している。

今回の場合パラメータは $\vec{J}$、J である。このハミルトニアンからボルツマン分布（式5.18）により次のように求められる。

ボルツマン分布

$$Q_J(d = 1|\vec{x}) = \frac{e^{\beta(\vec{J}\cdot\vec{x}+J)}}{1 + e^{\beta(\vec{J}\cdot\vec{x}+J)}} \Rightarrow \frac{e^{(\vec{J}\cdot\vec{x}+J)}}{1 + e^{(\vec{J}\cdot\vec{x}+J)}} = \sigma(\vec{J} \cdot \vec{x} + J) \tag{5.35}$$

$$Q_J(d = 0|\vec{x}) = \frac{1}{1 + e^{(\vec{J}\cdot\vec{x}+J)}} = 1 - \sigma(\vec{J} \cdot \vec{x} + J) \tag{5.36}$$

ここで導入した関数 σ をシグモイド関数といい非線形関数となっている。但し、β はパラメータ $\vec{J}, J$ にまとめることができるので以下では β は無視する。また出力の平均値は次の通り。

$$< d >= 1 \cdot \sigma(\vec{J} \cdot \vec{x} + J) + 0 \cdot \{1 - \sigma(\vec{J} \cdot \vec{x} + J)\} = \sigma(\vec{J} \cdot \vec{x} + J) \tag{5.37}$$

データ分布 $P(d,\vec{x})$ とのカルバック・ライブラー距離は

$$\begin{aligned} D(P\|Q_J) &= \sum_{\vec{x},d} P(d,\vec{x}) \log \frac{P(d,\vec{x})}{Q_J(d,\vec{x})} \\ &= \sum_{\vec{x},d} P(d,\vec{x}) \log \frac{P(d|\vec{x})P(\vec{x})}{Q_J(d|\vec{x})P(\vec{x})} \\ &= \sum_{\vec{x},d} [-P(d,\vec{x}) \log Q_J(d|\vec{x}) + P(d,\vec{x}) \log P(d|\vec{x})] \end{aligned} \tag{5.38}$$

以下 J に依存する第一項のみを取り扱う

$$\begin{aligned} &\sum_{\vec{x},d} -P(d,\vec{x}) \log Q_J(d|\vec{x}) \\ &\simeq \sum_{data:i} -\frac{1}{N} \log Q_J(d[i]|\vec{x}[i]) \\ &= \sum_{data:i} -\frac{1}{N} [(1-d[i]) \log\{1-\sigma(\vec{J}\cdot\vec{x}+J)\} + d[i] \log \sigma(\vec{J}\cdot\vec{x}+J)] \end{aligned} \tag{5.39}$$

但し、一つ目の $\simeq$ では真のデータ分布 $P(d,\vec{x})$ を N 個のサンプルの分布で近似した。$\vec{x}[i]$ は i 番目のデータの入力、$d[i]$ は正解の値である。

活性化関数

$$< d >= \sigma(\vec{J}\cdot\vec{x}+J) \tag{5.40}$$

誤差関数

$$L(< d >, d) = -(1-d)\log(1- < d >) - d \log < d > \tag{5.41}$$

以上より活性化関数としてシグモイド関数、誤差関数としてはクロスエントロピーが出てきた。どちらも実際に深層学習で使われることが多い。

5.3.3　多値出力

多値出力つまり入力 $\vec{x}$、出力 $\vec{d} = (0,\cdots,1,\cdots,0)$ のような場合を考える。例えば、$\vec{x}$ が画像、画像が猫の時は $\vec{d} = (1,0,0)$、犬の時は $\vec{d} = (0,1,0)$、鳥の時は $\vec{d} = (0,0,1)$ となるような場合がこれにあたる。この時次のようなハミルトニアンを考える。

ハミルトニアン

$$H = -(\mathbb{J}\vec{x} + \vec{J}) \cdot \vec{d} \tag{5.42}$$

このハミルトニアンは二値出力の場合とほとんど形が同じである。但し、$\mathbb{J}$ は行列を表す。この時、$\vec{d}$ の第 i 成分を d_i とするとボルツマン分布は次のようになる。

ボルツマン分布

$$Q_J(d_i = 1|\vec{x}) = \frac{e^{(\mathbb{J}\vec{x}+\vec{J})_i}}{\sum_j e^{(\mathbb{J}\vec{x}+\vec{J})_j}} = \sigma_i(\mathbb{J}\vec{x} + \vec{J}) \tag{5.43}$$

但し、ここで導入した非線形関数 $\sigma_i(\vec{a})$ をソフトマックス関数という。ソフトマックス関数は次のような性質を持つ。

$$\sigma_i(\vec{a}) \simeq \begin{cases} 1 & (d_i \gg d_j \quad for \quad \forall j \neq i) \\ 0 & (\exists j \quad s.t. \quad d_j \gg d_i) \end{cases} \tag{5.44}$$

i 成分が最大となるときだけ 1 となる連続関数という意味でソフトマックス関数という。出力の平均値については

$$< d_i >= \sigma_i(\mathbb{J}\vec{x} + \vec{J}) \tag{5.45}$$

データ分布とのカルバック・ライブラー距離のパラメータに依存する部分は

$$\begin{aligned} -\sum_{\vec{x},\vec{d}} P(\vec{x},\vec{d}) \log Q_J(\vec{d}|\vec{x}) &\simeq -\sum_{data:i} \frac{1}{N} \log Q_J\left(\vec{d}[i]\middle|\vec{x}[i]\right) \\ &= -\sum_{data:i}\sum_{k} \frac{1}{N} d_k[i] \log Q_J\left(\vec{d}[i]\middle|\vec{x}[i]\right) \\ &= -\sum_{data:i}\sum_{k} \frac{1}{N} d_k[i] \log \sigma_i\left(\mathbb{J}\vec{x} + \vec{J}\right) \end{aligned} \tag{5.46}$$

以上より

活性化関数

$$< d_i >= \sigma_i(\mathbb{J}\vec{x} + \vec{J}) \tag{5.47}$$

誤差関数

$$L(< \vec{d} >, \vec{d}) = -\sum_i d_i \log < d_i > \tag{5.48}$$

従って多値出力の場合、活性化関数としてはソフトマックス関数、誤差関数としてはクロスエントロピーが導出される。これらは実際に用いられることが多い。

5.3.4 連続値出力

次に連続値出力、つまり入力が $\vec{x}$、出力が実数ベクトル $\vec{d}$ の場合を考える。例えば、$\vec{x}$ が天気図の画像、出力 $\vec{d}$ が各地の予想降水量の場合がこれにあたる。連続値出力の場合、物理自由度が連続値となることから二値出力や多値出力のときのハミルトニアンと同じような形では状態が安定しないことに注意する。例えば次のようなハミルトニアンを考

える。

$$H = Jd \cdot x \tag{5.49}$$

ボルツマン分布では低温時は低エネルギー状態が支配的となるが、このハミルトニアンでの最低エネルギー状態を考えると Jx が正の時、$d \to \infty$ となり物理的意味をなさない。これは物理系に安定状態が存在しないために減校状態が実現できないからである。例えば無限に深い井戸を考えると粒子はずっと落ちていくために安定な状態を作ることができない。これを防ぐためには有限の最低エネルギー状態を持つような物理系を用意する必要がある。その中でも単純なモデルとして次のハミルトニアンについて考察する。

ハミルトニアン

$$H = \frac{1}{2}\{\vec{d} - (\mathbb{J}\vec{x} + J)\}^2 \tag{5.50}$$

このハミルトニアンは式 (5.5) の調和振動子のハミルトニアンと似ている。このハミルトニアンについてのボルツマン分布は

ボルツマン分布

$$Q_J(\vec{d}|\vec{x}) = \frac{\exp[\frac{1}{2}\{\vec{d} - (\mathbb{J}\vec{x} + \vec{J})\}^2]}{\int d\vec{d}\exp[\frac{1}{2}\{\vec{d} - (\mathbb{J}\vec{x} + J)\}^2]} = \frac{\exp[\frac{1}{2}\{\vec{d} - (\mathbb{J}\vec{x} + \vec{J})\}^2]}{(2\pi)^{\frac{n}{2}}} \tag{5.51}$$

この様に連続値出力の場合ボルツマン分布はガウス分布となる。出力はガウス分布の性質より

$$< \vec{d} >= \mathbb{J}\vec{x} + \vec{J} \tag{5.52}$$

またカルバック・ライブラー距離は次のように計算できる。

$$\begin{aligned} & -\sum_{\vec{x},\vec{d}} P(\vec{x},\vec{d}) \log Q_J(\vec{d}|\vec{x}) \\ & \simeq \sum_{data:i} \frac{1}{N}\left[\frac{1}{2}\{\vec{d}[i] - (\mathbb{J}\vec{x} + \vec{J})\}^2 + \frac{n}{2}\log 2\pi\right] \end{aligned} \tag{5.53}$$

以上の結果をまとめると

出力

$$< \vec{d} >= \mathbb{J}\vec{x} + \vec{J} \tag{5.54}$$

誤差関数

$$L(< \vec{d} >, \vec{d}) = -\frac{1}{2}\{\vec{d} - < \vec{d} >\}^2 \tag{5.55}$$

今回は活性化関数はなくただの線形変換が出力となっている。誤差関数としては二乗和誤差が導出された。これは線形回帰のモデルとして有名である。

5.4 深層化

以上より活性化関数と誤差関数が導出されたが、隠れた自由度を持つ系を考えることで深層化することもできる。隠れた自由度を持つ系とは出力としては機能しない物理的自由度のことである。例えば、原子は内部構造として陽子、電子、中性子をもちそれぞれが物理的自由度を持っているが、原子の速度分布を出力とするときはそれらの自由度は隠れた自由度となる。次に式を用いて説明する。

以下では外場 $\vec{x}$ がかけられた物理的自由度が $\vec{d}$ の物理系のハミルトニアンを

$$H_{\vec{x}}(\vec{d}) \tag{5.56}$$

と記述する。この時、次のようなハミルトニアン連なりを考える。

$$H_{\vec{x}}(\vec{h_1}), H_{\vec{h_1}}(\vec{h_2}), \cdots, H_{\vec{h_N}}(\vec{d}) \tag{5.57}$$

注意深く読めば、あるハミルトニアンの物理的自由度が次のハミルトニアンの外場として作用するようなモデルを考えていることがわかると思う。物理との対応を考えると $H_{\vec{x}}(\vec{h_1})$ は一つの原子の構造について、$H_{\vec{h_1}}(\vec{h_2})$ は原子同士の電磁気的な相互作用について、$H_{\vec{h_N}}(\vec{d})$ は原子からなる物体の重力相互作用についてのハミルトニアンだとして階層ごとに分解しているとみることができる。出力の確率分布は次のようになる。

$$Q_J(\vec{d}|\vec{x}) = \sum_{\{\vec{h_i}\}} Q_{J_N}(\vec{d}|\vec{h_N}) Q_{J_{N-1}}(\vec{h_N}|\vec{h_{N-1}}) \cdots Q_{J_0}(\vec{h_1}|\vec{x}) \tag{5.58}$$

$\vec{h_i}$ は隠れた自由度であることに注意する。また、$Q_{J_{i-1}}(\vec{h_i}|\vec{h_{i-1}})$ は $H(\vec{h_i}|\vec{h_{i-1}})$ によるボルツマン分布である。このままでは扱いにくいので次のような近似を考える。

$$\sum_{\vec{h_1}} Q_{J_1}(\vec{h_2}|\vec{h_1}) Q_{J_0}(\vec{h_1}|\vec{x}) \simeq Q_{J_1}(\vec{h_2}| < \vec{h_1} >) \tag{5.59}$$

この様な自由度についての足し上げを平均値を代入するような操作を繰り返すと次のよう展開できる。

ボルツマン分布

$$Q_J(\vec{d}|\vec{x}) \simeq Q_{J_N}(\vec{d}|<\vec{h_N}>) \tag{5.60}$$

但し、

$$\begin{aligned} <\vec{h_N}> &= \sigma(\mathbb{J}_{N-1}\cdot <\vec{h_{N-1}}> + \vec{J_{N-1}}) \\ <\vec{h_{N-1}}> &= \sigma(\mathbb{J}_{N-2}\cdot <\vec{h_{N-2}}> + \vec{J_{N-2}}) \\ &\vdots \\ <\vec{h_1}> &= \sigma(\mathbb{J}_0\cdot\vec{x} + \vec{J_0}) \end{aligned} \tag{5.61}$$

入力から出力までの変換が、線形変換と活性化関数による非線形変換の繰り返しになっていることがわかる。このように隠れた自由度を持つ系を考えることで深層構造を導入することができた。またカルバック・ライブラー距離を見ると

$$-\sum_{\vec{x},\vec{d}} P(\vec{d},\vec{x})\log Q(\vec{d}|\vec{x}) \simeq -\sum_{\vec{x},\vec{d}} P(\vec{d},\vec{x})\log Q(\vec{d}|<\vec{h_N}>) \tag{5.62}$$

つまり、深層化した際の誤差関数の形は最終出力層のみに依存する。

5.5　まとめ

以上より統計力学を用いることで活性化関数、誤差関数を導出し、更には深層化を導入することもできた。理論物理は自然界の現象を説明するために数理モデルを立てて議論・研究する学問である。ニューラルネットワークが脳内の神経回路の数理モデルとして作られたことを考えると、物理と深層学習は相性がいい。実際に物理的な観点から深層学習の振る舞いを理論的に解き明かそうとするような研究が盛んにおこなわれている。

昔、蒸気機関を研究することによって熱力学は発展していった。現在、量子コンピュータの研究をすることによって量子力学の情報理論的側面の理解が進んでいっている。深層学習について探求することで、また新しい物理が生まれるかもしれない。

参考文献

[1] 斎藤康毅,『ゼロから作る Deep Learning　—Python で学ぶディープラーニングの理論と実装』, オライリー・ジャパン,2016

[2] 高木貞治, 『定本 解析概論』, 岩波書店 2010

[3] Wikipedia, “Matrix calculus”, https://en.wikipedia.org/wiki/Matrix_calculus

[4] Yoshinobu Ogura (@wsuzume in Qiita), 『数学勉強ノート（偏微分と全微分）～～関数解析と多様体へ向けた微分の復習～～』, https://qiita.com/wsuzume/items/204a3defa86dace3e4b1

[5] Yu Umegaki (@AnchorBlues in Qiita), 『「ベクトルで微分・行列で微分」公式まとめ』, https://qiita.com/AnchorBlues/items/8fe2483a3a72676eb96d

[6] @takseki (@takseki in Qiita), 『ベクトルや行列を含む微分について』, https://qiita.com/takseki/items/b9a7115eb22040877922

[7] Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat’s striate cortex. The Journal of Physiology, 148(3), 574–591. https://doi.org/10.1113/jphysiol.1959.sp006308

[8] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4), 193–202. https://doi.org/10.1007/BF00344251

[9] Fukushima, K. (n.d.). A neural network model for selective attention in visual pattern recognition. 11.

[10] 人間の脳のメカニズムを、わたしは知りたくてたまらない。——福島邦彦 WIRED.jp. (日付なし). 読み込み 2019 年 11 月 20 日, WIRED.jp website: $https://wired.jp/waia/2019/09_kunihiko-fukushima/$

[11] Rescorla, R. A, & Wagner, AR. (1972). A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and non reinforcement. In AH. Black & W.F. Prokasy (eds.), Classical conditioning II: current research and theory (pp. 64-99) New York: Appleton-Century-Crofts.

[12] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning

applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[13] Amari, S. (1967). A Theory of Adaptive Pattern Classifiers. IEEE Transactions on Electronic Computers, EC-16(3), 299–307. https://doi.org/10.1109/PGEC.1967.264666

[14] Rumelhart, D. E., Hintont, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. 4.

[15] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[16] Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proceedings of the National Academy of Sciences, 111(23), 8619–8624. https://doi.org/10.1073/pnas.1403112111

[17] Simonyan, K., Vedaldi, A.,& Zisserman, A. (2014). Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. ArXiv:1312.6034 [Cs]. Retrieved from http://arxiv.org/abs/1312.6034

[18] Horikawa, T., & Kamitani, Y. (2017). Generic decoding of seen and imagined objects using hierarchical visual features. Nature Communications, 8(1), 15037. https://doi.org/10.1038/ncomms15037

[19] Shen, G., Horikawa, T., Majima, K., & Kamitani, Y. (2019). Deep image reconstruction from human brain activity. PLOS Computational Biology, 15(1), e1006633. https://doi.org/10.1371/journal.pcbi.1006633

[20] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. ArXiv:1409.3215 [Cs]. Retrieved from http://arxiv.org/abs/1409.3215

[21] Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural Machine Translation by Jointly Learning to Align and Translate. ArXiv:1409.0473 [Cs, Stat]. Retrieved from http://arxiv.org/abs/1409.0473

[22] Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. ArXiv:1508.04025 [Cs]. Retrieved from http://arxiv.org/abs/1508.04025

[23] Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2018). ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. ArXiv:1512.05193 [Cs]. Retrieved from http://arxiv.org/abs/1512.05193

[24] Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017).

Neuroscience-Inspired Artificial Intelligence. Neuron, 95(2), 245–258. https://doi.org/10.1016/j.neuron.2017.06.011

[25] 宅洋一郎. ゲーム AI 技術入門. 技術評論社, 2019

[26] 藤康毅. ゼロから作る Deep Learning 2. オライリージャパン, 2018

[27] tephen Grand, Dave Cliff, Anil Malhotla. Creatures: Artificial Life Autonomous Software Agents for Home Entertainment.http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.1278&rep=rep1&type=pdf . 1996 (最終閲覧日:2019 年 11 月 19 日)

[28] 見正雄, 飯島健太, 上田健次郎. ニューラルネットワークを用いた音声信号によるリップシンク (口パク生成) 技術. CEDEC2019. 2019

[29] eiichi Yokoyama. 「ニューラルネットワークを用いた AI の格闘ゲームへの組み込み」レポート [GCC2019].https://automaton-media.com/devlog/report/20190511-91689/ . 2019 年 (最終閲覧日:2019 年 11 月 19 日)

[30] ueki. ［TGS 2019］昔「テグザー」を作った人が機械学習で Oculus Go の 3DoF 入力を改善していた.https://jp.gamesindustry.biz/article/1909/19091202/ . 2019 年 (最終閲覧日:2019 年 11 月 19 日)

[31] ttps://www.ea.com/seed/news/self-learning-agents-play-bf1 . (最終閲覧日:2019 年 11 月 21 日)

[32] 伝的アルゴリズムによる人工知能を用いたゲームバランス調整. https://cedil.cesa.or.jp/cedil_sessions/view/1655. 2017 年 (最終閲覧日:2017 年 11 月 21 日)

[33] 着の持てる流暢な発話でゲーム業界に新風.https://www.toshiba-sol.co.jp/case/case2017/cpc.htm . 2016 年 (最終営業日:2019 年 11 月 21 日)

[34] 田中章詞, 富谷昭夫, 橋本幸士,『ディープラーニングと物理学』, 講談社,2019

[35] 阿部龍蔵,『統計力学（第二版)』, 東京大学出版会,2017

編集後記

今回は京都大学人工知能研究会会誌『Kaira vol.3』を購読して頂き、感謝申し上げます。この会誌は当サークルが発足した 2017 年以来毎年発行しているもので、人工知能の魅力と面白さを広く知ってもらうことを目的としています。

今年の会誌の内容を振り返ってみましょう。まず前半は深層学習の基本的、一般的事項とそれ理解するための数学的知識を紹介しました。続いて、後半は具体的・専門的な内容になっていき、その内容は「脳科学」「ゲーム」「統計力学」と多岐にわたります。

この後半部のジャンルの広さは人工知能が様々な分野とかかわりあっていること示すいい例だと思います。人工知能に対して様々なアプローチの方法があり、それ故に分野をまたいで活躍する。その意味で人工知能は多面多臂であり、どこまでも魅力的です。

そんな魅力の一端でも本会誌で伝われば幸いです。もし当サークルで一緒に人工知能について勉強したいという方がいらっしゃっいましたら、巻頭に連絡先が載っておりますのでご一報ください。

今回は購読して頂き、本当にありがとうございます。

京都大学人工知能研究会会員
会誌担当
田中大登

文責

はじめに　大山百々勢
第 1 章　深層学習入門　田中大登
第 2 章　機械学習のための偏微分入門　松原徳秀
第 3 章　脳科学から学んだ人工知能　野中聡馬
第 4 章　ゲームと機械学習の関わり　羽原丈博
第 5 章　統計力学から見る活性化関数と誤差関数　田中大登
編集後記　田中大登
冊子編集　田中大登