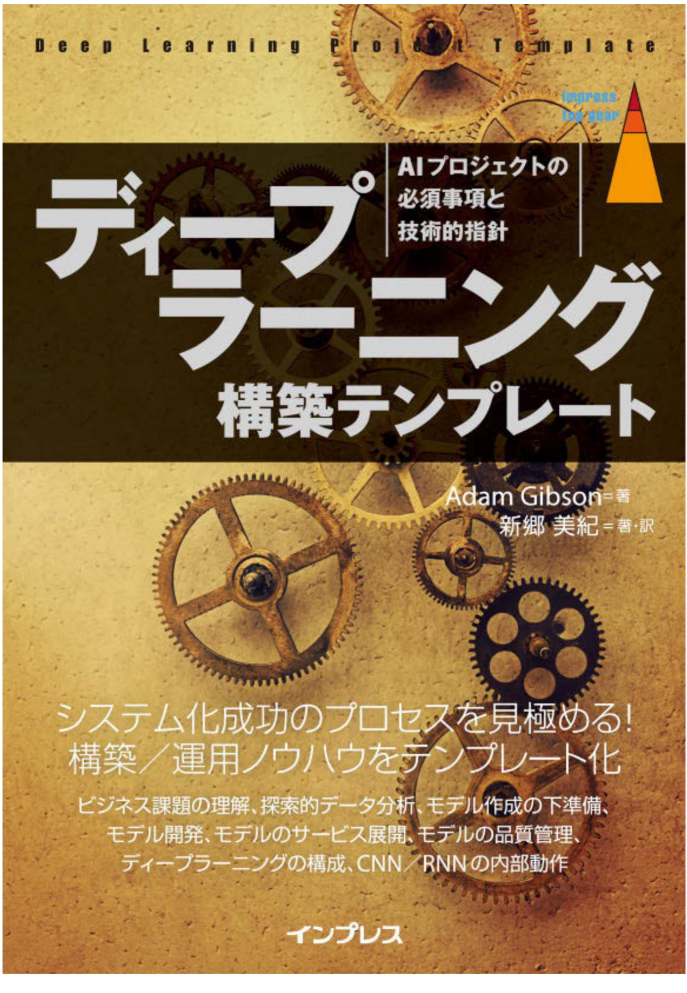

Deep Learning Project Template
AIプロジェクトの
必須事項と
技術的指針
ディープラーニング
構築テンプレート
Adam Gibson=著
新郷 美紀=著・訳
システム化成功のプロセスを見極める!
構築／運用ノウハウをテンプレート化
ビジネス課題の理解、探索的データ分析、モデル作成の下準備、
モデル開発、モデルのサービス展開、モデルの品質管理、
ディープラーニングの構成、CNN／RNNの内部動作
インプレス

■正誤表のWebページ、サンプルの入手先

正誤表を掲載した場合は、下記URLのページに表示されます。

また、本書のサンプルコードの入手先は、このページの「ダウンロード」欄に記載されています。

https://book.impress.co.jp/books/1119101017

※本文中に登場する会社名、製品名、サービス名は、各社の登録商標または商標です。

※本書の内容は執筆時点のものです。本書で紹介した製品／サービスなどの名前や内容は変更される可能性があります。

※本書の内容に基づく実施・運用において発生したいかなる損害も、著者、ならびに株式会社インプレスは一切の責任を負いません。

※本文中では®、™、©マークは明記しておりません。

まえがき

本書は、ディープラーニングの実装を現場で行っている方、ディープラーニングのプロジェクトの管理を行っている方、もしくはそのような予定がある方が対象です。ディープラーニングの基礎的な知識はある程度わかっている、書籍やWebサイトで公開されているサンプルプログラムなどは実行できる、といったレベルにある読者が、研究室や企業の現場で実践的な利用を進めるときに役立つ情報を提供することを目的にしています。

本書は、第一著者のアダム・ギブソン氏の発案によるものです。彼は、ディープラーニングの黎明期から、世界初となるオープンソースのJavaディープラーニングフレームワークDeepLearning4jを開発し、それ以降AIフレームワークなどの開発や、企業へのAIの採用の推進役を務めているのみならず、さまざまな講演や著作をはじめとするAIの普及に向けた活動においても第一線で活躍しています。

近年のAIブームの中で中心的なテクノロジーとしてディープラーニングが取り上げられ、企業などではデータ分析でディープラーニングを用いる機会が増えてきています。しかしながら、実際にプロジェクトとして成功しているものは一部にとどまります。

そうした中でAIのさまざまなプロジェクトで豊富な経験があるアダム氏と話をした際、ディープラーニングのプロジェクトが失敗しやすい原因がどこにあるのかについてアドバイスを受けました。彼によれば、その原因は、ディープラーニングの実装を成功に導くために必要な知識に触れる機会が少ないことにあるというのです。

そこで、日本のユーザ向けにこのような経験を書籍として形にするのはどうかという提案を後日、彼からもらい、インプレスに彼のアイデアを説明したところ、本書の出版について快諾をいただきました。

アダム氏は、大の日本好きです。もともとは米国で生まれ、大学在学中に米国

で起業されましたが、2015年からは日本を拠点に活動しており、日本のディープラーニングの発展に少しでも貢献できることに喜びを感じています。

本書は、基礎的なディープラーニングの知識はあるものの、実践でつまずいた経験があるエンジニアの方々、あるいは、ディープラーニングのプロジェクト管理を行っている、または組織の立ち上げに従事している、といった中で、どのようにプロジェクトを推進するべきかについて方針が立っていない方々に、何らかのヒントを与えられるような情報を提供します。本書を通じて、ディープラーニングをプロジェクトで扱う際の障壁が少しでも減り、プロジェクトの成功体験が増えることになれば、著者陣にとっては望外の喜びです。

最後に、本書の執筆にあたり、構成の段階から執筆さらには翻訳・校正で多くの方々にご協力・ご支援を賜りました。この場を借りて御礼を申し上げます。

2020年8月

新郷 美紀

CONTENTS

第1章

ディープラーニングプロジェクトはなぜうまくいかないのか

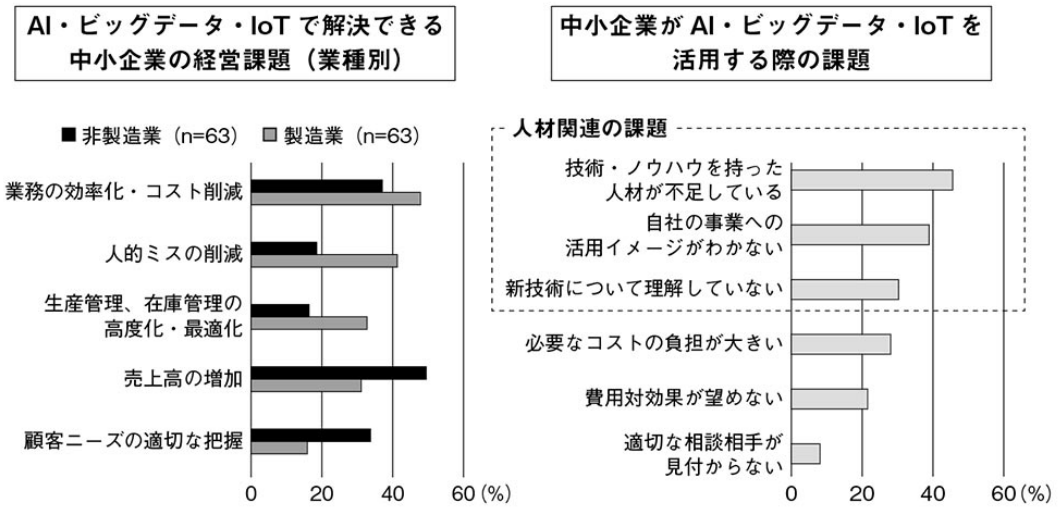
ディープラーニングの現状とテンプレートの意義

Deep Learning Project Template

近年、AI技術の急速な発展により、研究機関をはじめ企業などでAIが活用される機会が広がってきました。実際、皆さんがお使いのGoogle検索や翻訳機能、空港の顔認証ゲート、自動車の自動運転など、さまざまな分野ですでに活用されていることはご存じでしょう。

一方でAIを適切に扱って課題解決ができるような人材は現時点で不足しており、この状態はこれからも続くものと予測されています。

実際、2019年に経済産業省がAIの人材不足を解決するためのAI人材育成の試みとして、「AI Quest」という事業を立ち上げたことを記憶されている方がいらっしゃるでしょう。AIの人材育成は急務になっています。「AI Quest」では、年間2,000人のエキスパートレベルの人材を育成するという目標が掲げられていることからも、人材不足は深刻な事態になっていることがわかります。AI Questの資料内でも以下のように、AIの知識の理解および人材不足が課題となっています。



出典）中小企業の成長に向けた事業戦略等に関する調査（2016年11月）

図1-1:AI等を活用する際の課題―［出典］「AI Questについて」経済産業省 商務情報政策局

AI人材の育成について日本の現状を考えると、現在のAI人材育成プログラムや著作なども盛んで、さまざまな情報が提供されています。ただほとんどのものがAIを学んでいくうえで必要な理論や、サンプルプログラムを使った実装に重点が置かれているのが現状です。もちろんこのような知識は非常に重要で、実践でも役立つものです。しかしながら、それらの知識だけでは、デモを見せるためのプログラムは作れても、実際のAIの開発現場でうまくAIを活用できていないケースが多く存在します。サンプルプログラムを実装するレベルの知識が求められる場合の多くは、解くべき課題が与えられていて、さらにデータについても分析に値するものがあらかじめ取捨選択がされているため、データや課題の詳しい内容について自ら検討する必要がありません。そのような状態で実際のAIの開発プロジェクトに入った場合、プロジェクトを推進していくのに必要な知識が不足するため、戸惑われるのではないでしょうか。

本書の対象読者としては、以下のような方々を想定しています。

●データサイエンティストであり、作成したディープラーニングのモデルをアプリケー

ションに適用することや前処理などについて知識を深めたい方

- アプリケーションエンジニアやデータエンジニアなどのシステムエンジニアであり、ディープラーニングを学びたい方
- プロジェクト決定権を持つか、プロジェクトマネージャであり、ディープラーニングのプロジェクトでやるべきタスクがイメージできていない方

1.1 本書がディープラーニングに注力する理由

AIの活用が求められている現場において、独自でAIの課題やデータに向き合い、企業の中で本当にうまくいっている、という実例はまだまだ少ない印象です。

本書は、AIの中でも特にディープラーニングのテクノロジーを中心に、AIプロジェクトを成功に導くために必要な知識について説明します。本書でディープラーニングに注力する理由として、主に以下の2点が挙げられます。

第1に、ディープラーニングは、翻訳、顔認証、音声認識など、ありとあらゆる分野の分析で用いられる技術になっており、今後も利用分野が増えると期待されているテクノロジーだからです。近年のAIの発展は、ディープラーニングによる部分が大きいといっても過言ではありません。実際に、自動運転、映画のリコメンデーション、CT画像上での癌の部位の特定など、さまざまな分野で実用化されているか、今後の実用化が見込まれる技術となっています。

第2に、ディープラーニングは実践でつまずきやすいテクノロジーでもあるからです。ディープラーニングは、AIの中でも、特にある程度以上のボリュームのデータから反復的に学習を行うことにより、自動的にデータに含まれる特徴やパターンを見つけ出す技術です。つまり、ディープラーニングを実装するエンジニアは、特にデータに関して深い知識がなくても、真のチューニングによる精度向上を除けば実装はできてしまいます。そのため、商用を目的に実際に展開しようとする段階になって初めて、今までのディープラーニングの学習の経験が不足していることに気づくエンジニアの方がいらっしゃるというケースも見かけます。

1.2 なぜディープラーニングから学ぼうとするのか

AIを勉強される中で、ディープラーニングから学習される方が増えてきています。その理由は主に以下の2つの理由にあると考えています。

1. ディープラーニングの将来性と情報入手の容易性

現在、ディープラーニングはAIの開発において欠かせない技術になっています。画像認識や音声認識などでは、他の手法と比較して高い認識精度を示すだけでなく、人による認識の精度をも凌駕するパフォーマンスにまで進化してきています。そのため、主に画像認識、音声認識、自然言語処理、リコメンデーションなどの領域では、企業などでも積極的にディープラーニングを利用するケースが増えてきています。

また、さまざまな機械学習に関する入門書やブログなどでも、ディープラーニングを中心に説明するものが多く見られるようになってきているので、AIの勉強をディープラーニングから始める方が増えてきています。

2. ディープラーニングは入力データの特性の理解を省略できてしまう

初めて機械学習を勉強する方にとって、ディープラーニングが始めやすい状況になっているもう1つの理由は、入力データに関する深い知識がなくてもスタートしやすいからでしょう。つまり、他の機械学習では、エンジニアが特徴量を指定することで、モデルがそのデータに含まれる特徴を抽出する必要がありますが、ディープラーニングでは、エンジニア自体はデータの特徴量を指定する必要がなく、モデルが自動的に入力データから特徴を抽出できてしまいます。そのため、入力データさえきちんと準備できれば、簡単に分析を始めることができます。つまり、データを扱うための細かい知識がなくても始めることができる傾向にあります。

なぜディープラーニングが他のテクノロジーと比べて始めやすいかについて、

具体的に説明しましょう。

たとえば、メールのシステムで、迷惑メール（スパムメール）か否かを判定するプログラムの作成を依頼されたとします。実際にこのような仕組みの実装に機械学習が必須というわけではなく、以下のような従来型のプログラミング手法でも実現可能です。

◉――従来型プログラムの実装フロー

1. いくつかのスパムメールの中身を人手で把握
   その中身に共通する特徴をエンジニア自身が特定する。
   例：
   - 広告メールであることが大半
   - HTML型のメールが多い
   - ショッピングサイトへ誘導するタイプの内容が多い
     （特に、健康食品、イベントチケット、アダルト商品など）
2. 1で特定できた特徴を検出できるようなアルゴリズムを開発し判別精度をチューニング
3. 1と2の作業を繰り返し、商用に耐えうる精度になったらアプリケーションに実装

この従来型に対して、ディープラーニングで同様のプログラムを実装する場合、以下のアプローチをとります。

◉――ディープラーニングでの実装フロー

1. メールのデータから、スパムか否かの成否判定用フラグを付与したデータに多量に変換して、データを準備
2. 1で準備したデータを入力として、成否など二値の判別で一般的に用いられるディープラーニングのアルゴリズムの1つを選択して訓練と開発を行い、判別精度をチューニング

3. 1と2の作業を繰り返し、商用に耐えうる精度になったらアプリケーションに実装

以下の図で見ていただくとイメージがわきやすいかと思います。

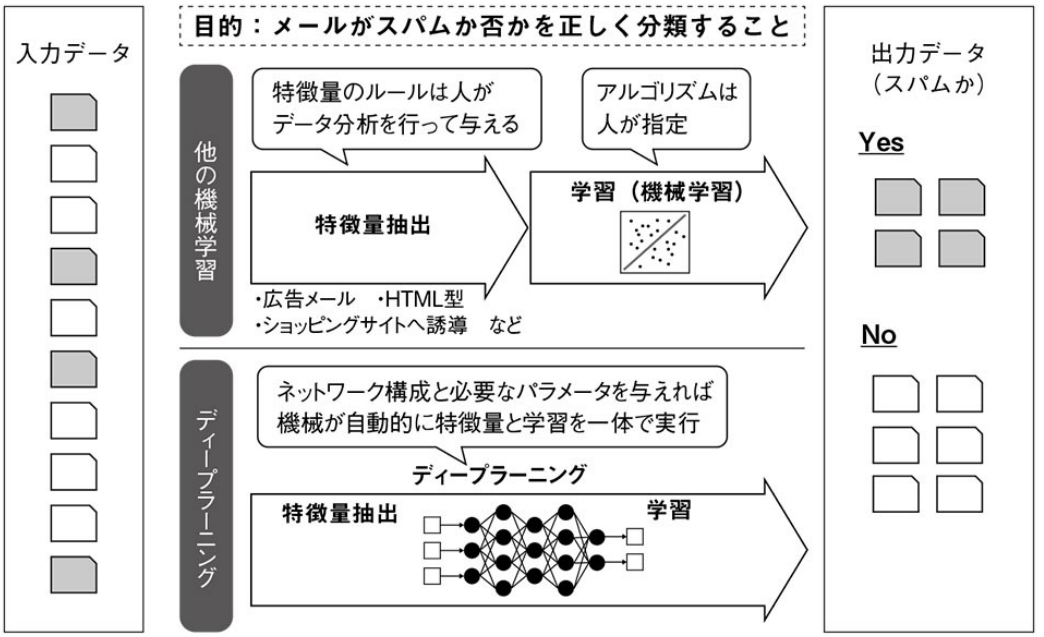


図1-2:他の機械学習とディープラーニングの違い

上記の違いでわかるように、従来型のプログラミング手法では、エンジニア自身がスパムの特徴を正確に把握し、その特徴を抽出するためのアルゴリズムをプログラミングすることから始めます。それに対して、ディープラーニングでは、上記の例のように教師ありのデータを用いる場合、成否判定用のフラグが付いたデータを準備すれば、判別用のルールは機械が自動で学習していきます。つまり、細かいルールを逐一、人手で実装しないアプローチになります。

ただし、実際のディープラーニングでプロジェクトを成功に導くには、データに関する知識は非常に大切なものです。特に分析結果が期待どおりのものになっているか否かは、人が最終的に判断する必要があります。つまり、用いたデータをはじめ分析結果に至るまでの妥当性を判断する役割は、設定した課題に関して専門的知識を持つエキスパートが担います。そのため、エンジニアの方々は、データに

ついての知識も求められることを十分理解しておいてください。

1.3 ディープラーニングが採用される要因

実際に企業などでディープラーニングの採用が進んでいる理由の1つについて、先ほどの迷惑メールをもとに説明します。

スパムメールは日々進化しているので、新しいスパムメールを実際に受け取られる方もいるでしょう。その最新スパムに対応するには、新たなスパムのルールを見つけ出して実装する必要があります。スパムメールは日々進化して新たな手法が生み出されています。スパムを送る側も、スパムと判定されるロジックを巧妙にかいくぐるように開発し実装してきます。その日々の進化にタイムリーに対応しようとした場合、人手で新たなスパムのロジックを見つけ出し、それを抽出するプログラムを開発するには多大な時間を要するため、対応が遅れがちになるでしょう。それに対し、新たなスパムメールに対してスパムを示すフラグを付与してディープラーニングのアルゴリズムに学習させることで、判定精度を上げることが可能であるため、より柔軟かつタイムリーに判別ができるようになります。

つまり、日々利用環境が変化する中でも柔軟に分析の精度を保つために、ディープラーニングの採用を求める企業も増えてきています。

また、ディープラーニングがここ数年でより多くの企業などで用いられるようになった要因として、上記以外にも主に以下の3点があります。

- 学習に利用するデータを準備しやすい環境が整った
- GPUに代表される高性能なコンピュータ環境が整った
- PyTorch、TensorFlowに代表される、一般のエンジニアが汎用的に利用可能なディープラーニングのフレームワークがそろった

現在はディープラーニングを利用するモチベーションも高くなり、利用に適した環境がそろいつつあります。その中にあり、企業などでディープラーニングの採用

に前向きになっている一方で、ディープラーニングを実行するための管理者やエンジニアの準備が不十分であることが、水を差しかねない状況にあるのも事実です。そのため、本書では機械学習の中でも特にディープラーニングに注力して、AIプロジェクトの成功確率を少しでも増やせるように、必要な情報を提供していきます。

1.4 ディープラーニングの現場での課題

上記のような柔軟性や利用環境の観点から、ディープラーニングが非常に便利なツールになってきています。ディープラーニングでは、スパムか否かについての詳細なルールをエンジニア自体が理解する必要もなく、さらにはその条件を正しく抽出するためのロジックを自らプログラミングする必要もないため、簡単に誰でも試せるものになっています。さらに、ディープラーニング入門用コンテンツを提供する多くのWebサイトなどでは、簡単にプログラムが試せるようになっています。そこでは、さまざまなお膳立てが事前に行われており、スパムを判定するか否かというプロジェクトのそもそもの動機付けから、その課題に必要なデータの準備（スパムとスパムでないデータの準備、さらには、スパムか否かのフラグ付けまで行ったデータセットの準備）に加え、課題に適したアルゴリズムの選択まで用意されています。このように、プロジェクトの動機付けからアルゴリズムの選択まで学習する必要がないWebサイトが多くなっています。そのようなサイトで学んだ後では、実装経験をあまり積んでいないエンジニアの場合、そもそも課題設定ができず、さらには必要となるデータの準備もままならず、選択すべきアルゴリズムもわからないといった状況になります。

これはエンジニアのスキルアップという点でも、諸刃の剣です。簡単に始められて、しかもエンジニア自身がデータに対して知識を十分に持たなくても分析結果を出せるようになります。ディープラーニングでは、精度などは別にしても何らかの分析結果を出すのは、他の機械学習よりもさらに簡単です。他のテクノロジー

よりデータなどに関する知識が不十分なままでも、ディープラーニングの勉強はある程度進めていくことが可能です。そのため、簡単なサンプルコードを用いてディープラーニングのモデルを作成できるエンジニアの数は増えています。これは、ディープラーニングの普及という点では貢献をしています。一方で、不十分な知識でもモデル自体は作れてしまうものの、精度などの点で企業が期待するだけの成果を出すには至らず、継続的な利用に否定的になる企業などが増えていく可能性があります。

このような知識不足が、ディープラーニングを現場で用いる際の落とし穴になっています。実際にディープラーニングのアルゴリズムを用いてモデルをチューニングする作業は、次の図1-3からわかるとおり、プロジェクト全体の作業の3割にしかならず、それだけの作業ではプロジェクト全体を進めていくことができません。機械学習を用いた分析においても、適切な課題の抽出から機械学習で利用可能となるデータの準備までは、非常に地道な作業となっています。しかしながら、期待する課題の達成に必要な条件を整えるには、こうした地道な作業でのトライアンドエラーによる学びが必要になります。

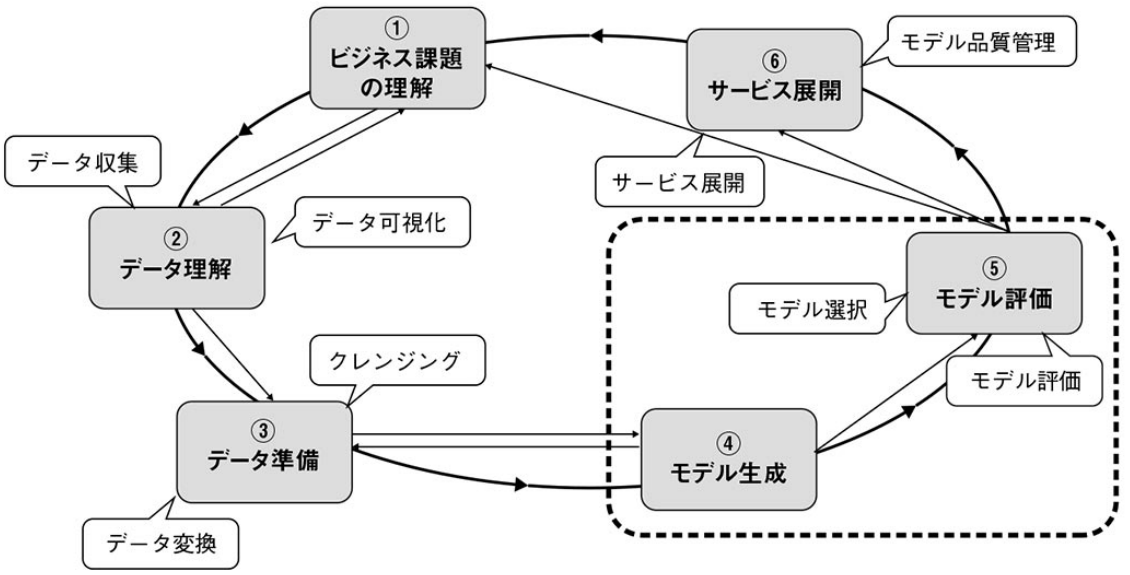


図1-3:プロジェクト全体のうちのモデル構築・チューニング作業の割合

また、不得意な課題をディープラーニングに適用してしまうというケースも見受けられます。スパムメール自体に学習用データがほとんど含まれていない場合、少ないサンプルデータを使って学習を進めていくことになります。このような状況は、人の思考パターンによってロジックを組み立てることで対応可能ですが、ディープラーニングを利用すると、サンプル数が少なすぎて精度が上がらないことがよくあります。

ディープラーニングでは、スパムか否かを自動で判別する仕組みを利用するため、スパムと判断した理由について開発者自体が説明できなかったり、多量のデータがない場合には、人手で実装した場合よりも精度が悪かったりするケースがあります。そのような状況を見てディープラーニングに失望する企業もあります。もともとのプロジェクトが取り組むべき課題としてディープラーニングが不向きである場合にも、ディープラーニングを当てはめているケースが少なくありません。このような状態に陥らないようにするためにも、プロジェクトを始めるにあたっては、ディープラーニングを使うのが適切かどうかを考慮する必要があります。

このように実際に現場でディープラーニングを用いるか否かについては、採用する前にさまざまな検討を行い、適切に判断していく仕組みが必要となります。

1.5 AIテンプレートの重要性

ここまでの説明でわかるように、ディープラーニングのプロジェクトを成功に導くには、さまざまな項目について考慮する必要があります。

読者の皆さんの中には、まだ実際に現場でディープラーニングの実装などを経験されたことがない方もいらっしゃると思います。実際に現場経験がある方であっても、特に実装の手法が確立していない中で、手探りで業務を完遂された方や、プロジェクトをリードされた方もいらっしゃるでしょう。

さらに、現場からの声として「ディープラーニングのプロジェクトの進め方がわからない」とか「導入から検証（PoC）までは実施したものの効果が出なかった」

という話を聞く機会が多くなっています。本書では、そのような方々がプロジェクトを少しでも前に進められるように、指針となる体系的な情報を提供します。

具体的には、著者の過去のプロジェクトでの成功もしくは失敗した場合の特性を分析し、同じ失敗を繰り返さないようにするためのノウハウ集を準備しています。このようなノウハウ集をAIテンプレートと名づけて活用することにより、経験の蓄積を継続していくようにします。AIテンプレートとは、ディープラーニングのプロジェクトを推進するうえで、道しるべとなる役割を果たすノウハウ集であり、著者の独自の経験に基づいて標準化された、ディープラーニングのプロジェクトで利用可能なテンプレートを指します。このテンプレートは、技術的な観点から説明するだけでなく、必要な手順によって与えられるビジネスへの影響についても示しています。ディープラーニングを用いたプロジェクトの適切な運営は、テクノロジーの側面のみならず適切なビジネス戦略に基づいて行われる必要があります。そのため、技術だけではなくビジネスも含めた視点を考慮した俯瞰的なテンプレートが役に立ちます。

本書では、テンプレートの作成にあたり、著者のアダム・ギブソン氏の豊富な経験をベースにしています。彼のこれまでのグローバルでのディープラーニングプロジェクトの実体験が詰まったテンプレートは実践的であり、ディープラーニングのプロジェクトの開発を請け負うエンジニアの視点のみならず、ディープラーニングのプロジェクトを率いる管理者の視点についても参考となる情報が盛り込まれています。

このような視点を中心にして記載された書物はほとんど存在せず、しかも本書はグローバルでの豊富な経験に基づいており、ぜひ読者の皆さんがディープラーニングプロジェクトに取り組む際に参考にしていただきたい内容となっています。

1.6 本書で取り上げる内容

本書では、ディープラーニングプロジェクトを成功に導くために必要な知識とし

て、すでに入手可能で一般的な情報を補完する方向性をもって、必要な情報を説明していきます。ディープラーニングのプロジェクトを採用すべきか否かの検討から始め、検討すべき内容とその指針となるAIテンプレートについて説明します。その後、ディープラーニングの基礎的なアルゴリズムを技術的に実装するにあたって参考になる情報を説明します。最後は、仮のユースケースをもとにディープラーニングプロジェクトの進め方を疑似体験できる情報を提供します。

具体的には、第2章では、機械学習（特にディープラーニング）のプロジェクトが成功しない理由から取り上げ、成功に導くためのAIテンプレートの導入について説明します。この章は、機械学習プロジェクトの技術的な側面のみならず、ビジネス的な側面で検討すべき内容も含むことにより、企業等でプロジェクトの導入を検討されている技術者、さらには管理者の方にも有益となる情報を提供するようにします。

第3章から第5章では、ディープラーニングの技術的な側面を中心に説明します。技術的側面に関しては、他の多くの入門に関する情報でしばしば省略されがちな、ディープラーニングの基本と、一部の高度な概念をカバーします。アーキテクチャの観点から、モデル構築ステップの詳細について説明している本は他にも数多くあるので、同様の情報をあまり繰り返さないようにしています。その代わり、ディープラーニングのモデル開発において、モデルの精度や性能を一定程度チューニングするための1つの指針を入手いただけるように、テクノロジーの核心部分についていくつか直感的な説明をしていきます。

まず第3章では、ディープラーニングの基礎的な知識について、より直感的な理解を深められるようにします。その中で、ディープラーニングのアルゴリズムの選択や、チューニングを行う際に1つの指標となる情報を提供します。

第4章では、ディープラーニングの代表的なアルゴリズムで最もよく利用されている畳み込みニューラルネットワーク（CNN）について、直感的に理解しておくべき内容を説明します。

第5章では、CNNの次に広く利用されている再帰型ニューラルネットワーク

（RNN）について説明します。

ディープラーニングのアルゴリズムはCNNやRNNだけではなく、さまざまなものが開発されています。実際、日進月歩で新たなアルゴリズムが発表されています。本書でそれらをキャッチアップすることは不可能ですが、読者の方々がそれらのアルゴリズムを理解していくうえでも本書の内容が参考になるものと期待しています。

そして最後の第6章では、それまでの基礎的な内容を理解したうえで、1つのユースケースに基づきAIテンプレートを用いた開発／実装をイメージトレーニングできる内容にしています。

第2章

機械学習プロジェクトの標準プロセス［課題理解からメンテナンスまで］

AIテンプレートでミス／ギャップを解消する

Deep Learning Project Template

機械学習のプロジェクトを実行する場合、機械学習の問題に対してどのようにアプローチするべきかの検討から始めることになります。その際、さまざまなトレードオフが存在するため注意が必要です。チーム内で最初に課題もしくは基本的なユースケースを検討するときには、実運用で必要となる項目が省かれる傾向にあります。たとえば、実社会でモデルを利用し続けるのに必要となる堅牢なシステムであること、という項目です。このような要件は、手順から省かれてしまいがちです。その手順はモデル構築の部分に限らず、アプリケーションの他の部分でも発生する可能性があり、最終的にはプロジェクトの成功のカギを握る1つの要因になります。

この手順を省略してしまうと、チームはモデルの実証実験の段階から抜け出すことができなくなってしまいます。たいていの場合、システムを堅牢なものにしておかないと、プロジェクトの停滞が起こり、サービス展開ができなくなります。さもなければ、モデルがサービスとして展開できたとしても、サービスを利用するのに非常に時間がかかったり、セキュリティなどの問題が解決されていなかったりと、のちのちに問題が発生する場合が多くなります。その結果、プロジェクトの推進者をはじめとするステークホルダーは、機械学習が効果的なツールであるということに懸念を抱くようになり、最終的に失望してしまいます。いまだに多くの企業は、機械学習を活用するために必要な準備が十分に整えられておらず、導入時期としては早すぎる傾向にあります。これらの手順の検討が十分に行える状態になる前に機械学習を導入してしまうと、企業の管理職は、機械学習が単なる誇大広告なのか、

真にビジネス価値を生み出すものなのかがわからない状態に陥ることになります。

AIを用いたプロジェクトを成功に導くには、AIを実行していくプロセスを理解しておくことが重要です。

AIの処理フローでは、基本的に、データ分析で一般的に用いられる処理の概念が利用できます。データ分析の処理フローにはいくつかの方法論が存在しますが、本書では、代表的な処理フローのCRISP-DM（図2-1）に沿って説明していきます。**CRISP-DM**（CRoss-Industry Standard Process for Data Mining）は、同名のコンソーシアムによって提唱されたデータ分析プロジェクトのプロセスモデルです。図2-1からわかるように、データ分析では、ビジネス課題を理解することからスタートし、サービス展開に至るまでをフロー化しています。AIのプロジェクトを成功に導くには、目標となる機能や品質の実現、維持管理のために、各処理プロセスの試行錯誤を繰り返しながら最終的なサービス展開に持っていき、さらには維持管理や改善を継続していくために、このフローを定期的に繰り返していくことが重要です。

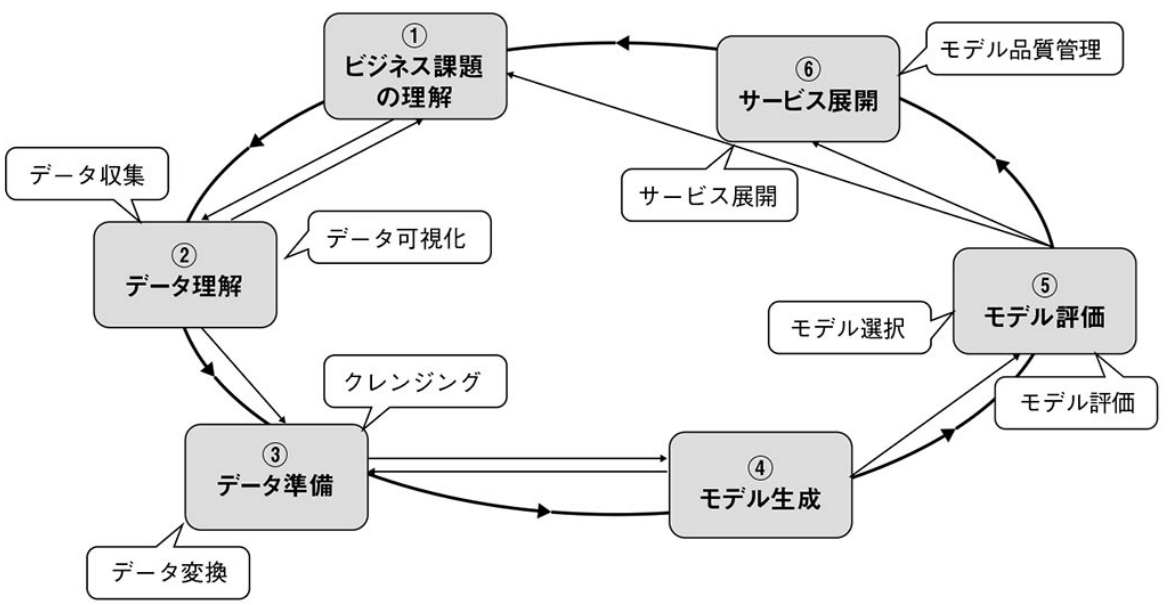


図2-1：AI処理フロー概要（CRISP-DM）

本章では、機械学習を導入する際に陥りやすいミスを避けながら、より多くのアプリケーションを早期に商用サービスとして投入できるようにすることを目的としています。そのために、手順を踏みながら実証実験と商用サービスとの間に生じるギャップを早い段階で特定し、そのギャップを埋めていきます。

2.1 ビジネス課題の理解

最初に解決すべきビジネス課題を理解し、課題解決に必要なさまざまな要件を明らかにしていくことが重要です。ビジネス課題の理解は、図2-2の破線枠で示したように、AI処理フローで最初に検討すべき内容になります。

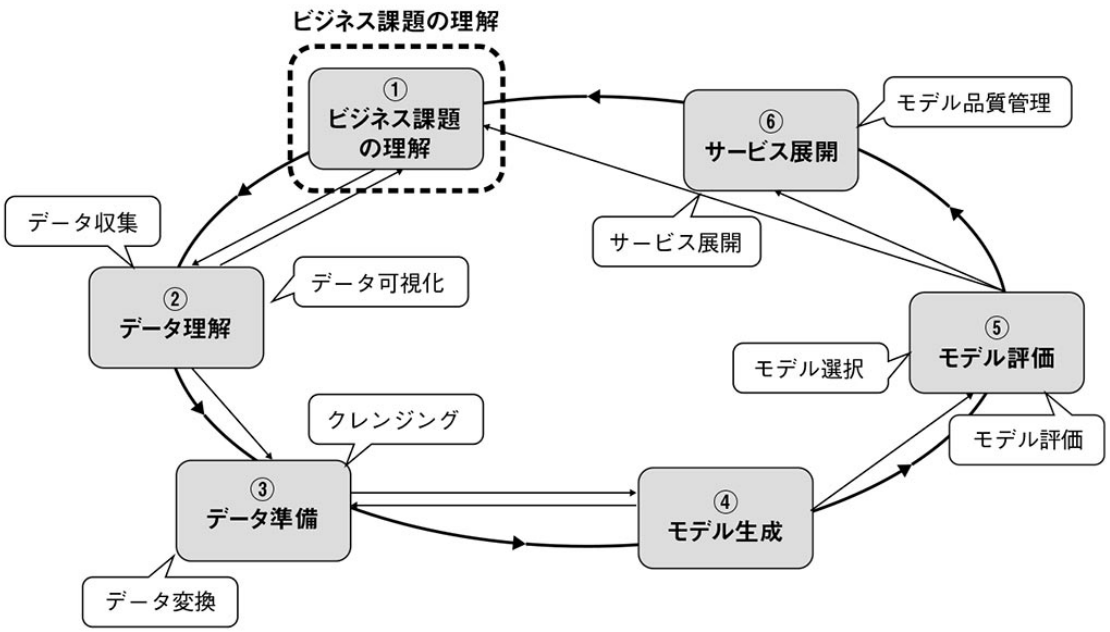


図2-2:ビジネス課題の理解

通常のCRISP-DMでは解決すべきビジネス要件の定義がメインに語られますが、実際にビジネスの課題を解決していくには、チームのスキル要件やITインフラ要件などを満たす必要があり、ビジネス課題を理解する段階でこれらの要件を検討しておくことをお勧めします。

▶2.1.1 チームのスキル要件

AIのプロジェクトを完遂するには、以下のようないくつもの異なるスキルセットを持ち合わせる複数のメンバーでチームを構成する必要があります。

1. （特に業種特有の問題に関する）業種の専門知識を持つエキスパート（ドメインエキスパートと呼ばれます）。プロジェクトでの役割は「目的の定義」
2. （アプリケーションの開発を担う）ソフトウェアエンジニア。プロジェクトでの役割は「モデルを使用したアプリケーションの構築支援」
3. （Pythonと、ある程度の統計学とデータ分析の知識を有する）モデル構築のエンジニア。プロジェクトでの役割は「ビジネス目標に合致したモデル構築」
4. （インフラストラクチャやビッグデータのエコシステムの知識を有する）データエンジニア。プロジェクトでの役割は「データ蓄積用のインフラ管理支援」

また、サービスを商用で提供する場合には、チーム内にプロダクトマネージャを置くのが一般的です。ビジネス要件は実際に解決したい課題によって異なるため、組織内で十分な知識を有するエンジニアが準備できない場合があります。そのような場合、プロダクトマネージャは、データサイエンティストを一時的に雇ったり、コンサルタントに具体的な問題に取り組んでもらうように依頼したりする場合があります。また、あるときには、チームのビジネス要件はチーム内で明確になっているものの、実際に機械学習を実装できるだけのスキルを持っているエンジニアがいない場合もあります。そのような場合には、たとえば、ユーザがいつサービスの利用を終了するかという解約分析のようなものや、さらに複雑なユースケース（後述）などの具体的な機械学習のモデルの実装を支援してもらうために、機械学習の分析エンジニアだけを雇うかもしれません。

▶2.1.2 適切なチームの構築方法

適切なチームは、上述のような異なるスキルセットを有する要員で構成される必

要があります。インテリジェントなアプリケーションを構築しようとする際、実際のソフトウェアをサービス展開する生産性の高いチームでは、データサイエンティスト専用のチームを構築するのではなく、各チームにデータサイエンティストが1名以上所属するような形式が望まれます。

なぜなら、以下のことが求められるからです。

1. ドメイン知識を有するエンジニアにより、現実的なユースケースに基づいて機械学習のモデルを構築可能になること
2. アサインされたデータサイエンティストが長期にわたってチームで働くことにより、ドメイン知識を学ぶことが可能になること
3. データサイエンティストと協働することで、アプリケーションに機械学習を適用するための製品のライフサイクル全体を管理できるようになること
4. データサイエンティスト専用のチームで構成された場合に陥りやすい（実験が多すぎるとか、ドメイン知識が不足しているなどの）共通のミスを避けることが可能になること
5. 連携によりチームの生産性が向上するため、多くの重複した会議を避けることができるようになること
6. 定期的に適切なフィードバックを行える体制が出来上がるので、実際の製品のライフサイクルに連動したアプリケーションに機械学習モデルを組み込み、維持することが可能になること

▶2.1.3 データサイエンティストを雇うタイミング

一般的には、実際に分析するのに十分なデータを有するチームが形成されてから、データサイエンティストを雇うべきです。データを持たない現場でデータサイエンティストを雇うのは生産性が良くありません。ときどき企業の中には、データの準備に先行してデータサイエンティストを雇うことが見受けられますが、結果的にチーム全体もしくは個人が解雇されて終わってしまいがちです。

チームが最初にデータサイエンティストを雇った時点では、データサイエンスの具体的な仕事が明確になっていないかもしれません。実際にそれ自体は問題ありません。もしかすると読者の方の中には、企業にとって最初に雇われたデータサイエンティストの人もいらっしゃるでしょう。そのような方の場合、チームの原動力としての役割も期待される場合が多く、それらの項目についてもカバーしておくことが非常に重要です。

もしあなたが課題解決に向けての原動力としての役割を期待されている人材の場合、解決すべき課題として、上記のようなものも含まれることを理解できている必要があります。インフラの準備ができていない段階で人を雇ってしまった場合、時間を無駄にしてしまうことになるので、インフラが整備されているかを確認してください。また、そのチームが何をしようとしていて、最初のいくつかのプロジェクトがどのようなものなのかについても、お互い正しく理解できていることをあらかじめ確かめておくようにしましょう。これにより、将来的に悪い誤解が発生するのを防ぐことができます。

データサイエンティストを雇うべきタイミングについては、次のようなケースを類推することで理解しやすくなるでしょう。製品販売のケースを考えてみると、営業職の人を雇う前に、しっかりとした売れる製品を作るのは当然のことです。このことを機械学習のタスクに置き換えてみると、データサイエンティストを雇う前に、データ分析に必要な環境の整備ができていることが重要になります。

▶2.1.4 ビジネス要件

ビジネス要件の検討には非常に広範囲の内容が含まれますが、大半はプロジェクトごとに個別に検討すべき内容です。ここでは、検討が広範囲になりがちなビジネス要件を少しでも焦点を絞り込めるようにするためのヒントについて説明します。

一般的にモデルを構築する場合、正しいチームの基準やビジネス要件に到達するまでに、いくつものプロトタイプを構築することになります。ビジネス要件は、モ

デルを実装するアプリケーションに対してモデルが与える影響の部分で定義されます。ビジネス要件として定義される課題としては、たとえば「さまざまな異なる種類の物体を特定する必要がある物体認識の課題」や「何らかの機械の障害を予測する課題」というようなレベルのものです。これらはいずれも扱われる課題の違いによって要件が大幅に異なり、1つの規則ですべてのケースをとらえることはできません。利用目的や入力データなどが明確になっている場合には、機械学習のモデル上に入力データをマッピングするところから検討を始めることも可能ですが、一般的には、ビジネス要件から検討を始めることがより汎用性が高い手法になります。

▶2.1.5 IT インフラ要件

以下の例は、カメラを使用して機械学習モデルが物体検出を行うユースケースです。一般的にはこのような場合、リアルタイムで物体を検出する必要があるため、機械学習モデルの開発だけでなく、たとえば以下のような項目を検討する必要があります。

1. （モデルを利用するまでに要する時間の）遅延
2. （ハードウェアの要件で決まる）メモリ容量
3. （ネットワークカメラで一般的に用いられる）画像のカメラからアプリケーションへの送信方法

カメラを使用した物体検出のユースケースの検討を進める際には、モデルの構築以外にも上記をはじめとするさまざまな要求事項を明確化しておく必要があります。要求事項はできるだけさまざまな視点で配慮するべきで、チーム全体で定義するべきです。

物体検知のモデルを構築するデータサイエンティストは、構築されたモデルが期待される精度を達成するために、さらにはカメラの操作性などのその他の要求

事項についても満たしていることを保証するために、ソフトウェアエンジニアのほか、このユースケースではカメラに関するドメインエキスパートと一緒に働く必要があります。これらの要求事項は、利用するカメラの正しい利用経験を持つエキスパートの知識が必要であり、彼らが定義すべき内容です。たとえば今回のユースケースで、カメラを用いてリアルタイムに物体の検知を行う必要があるとします。そのような場合、物体検知がカメラやカメラと連動する分析システムの基本性能として、リアルタイムに反応できるだけのシステム設計が必要になります。そうするためには、カメラやカメラと連動する分析システムで映像を十分に蓄積するためのメモリなどのリソースを有しているだけでなく、リアルタイムに物体検知の結果を返せるような仕組みとなっていることも考慮する必要があります。

この機能を実現するには、一般的に、物体検知のモデルをリアルタイムに耐えうる速度で動作させることはもちろん、そのほかにカメラからその物体検知のモデルを動かすシステムに対してカメラで撮影した映像がリアルタイムで取り込まれるようにすることを検討する必要があります。

一般的にカメラと異なるリモートの分析システムでは、物体検知のモデルの学習または推論を実行します。一方、カメラはリモートのネットワーク経由で、撮影した映像を分析システムの共有リソースにアップロードします。このような場合、カメラ自体の仕様として、映像をどこかのリモートのシステムにアップロードするために、ネットワークカメラのようにカメラ自体にWi-FiまたはLTE（携帯電話の通信規格）を使用できる機能を有するものを選ぶ必要があります。

以上の例からわかるように、物体検知のようなユースケースでも商用利用などを想定した場合には、機械学習のモデルの構築にとどまらず、より広い要件をできるだけ正確にとらえておくことが重要です。

また、大量のデータを蓄積してきている企業は、一般的にそれらのデータから価値を生み出して事業につなげようとしています。これは、製造業からSaaS（Software as a Service）のような企業まで、あらゆる分野に及んでいます。

データサイエンスやデータから価値を創出する準備ができている企業は、一般

的に**ソフトウェア・ネイティブ**と呼ばれています。

このような企業は、データの価値創出に有用なソフトウェアの構築方法を理解できる熟練のソフトウェアチームを持っているため、データ活用を直接ビジネスの利益の源泉につなげることができます。

そのような企業では、まずデータをさまざまな場面で統一的に利用できるように、インフラの準備を行うことが非常に重要です。一般的にデータをアプリケーションで利用する場合も機械学習に利用する場合も、同じデータ形式でデータを蓄積しましょう。また、データは何らかの方法で管理してバックアップするようにしましょう。それらのデータ量が十分にあれば、機械学習でも利用できます。そうした環境が整っていれば、機械学習のようなパターン認識を使って課題を解決するのに理想的です。

データ分析において、データ量が少なかったり、データの特徴量が少なかったりする場合には、探索的データ分析（EDA）の手法を用いて、データのルールを手動で見つけ出す必要があります。それに対し、ビッグデータと呼ばれるようにデータが多量にあったり、特徴量が非常に多かったりする場合には、人手でデータの特徴を認識することは不可能であり、機械学習の出番になります。通常、決められたルールの組み合わせで特定の問題を記述できない場合に、機械学習の利用を開始したいと思うのは自然の流れです。

大量のデータを蓄積している企業では、過去に複数のビジネスアプリケーションの構築経験があるだけでなく、効率化のために用いられている社内ツールから、外部ユーザが利用する有償のアプリケーションまで、さまざまなものを利用しています。さまざまなアプリケーションを効率的に実現するためのITインフラには、一般に以下のような仕組みが必要になります。ただし、以下ではすべての用例を網羅しているわけではありません。

1. アーカイブやバックアップ用の複数のデータベースの構築
2. 継続的なソフトウェアのテストとサービス展開に関するシステム統合

3. （基本的にプロダクトマネージャによって管理される）新たな機能に関する汎用的な承認の仕組みの構築
4. ソフトウェアバグに関する問題の追跡手法の確立
5. （機械学習では特に重要となる）ユーザからの継続的なフィードバックを得られるようなモデルの開発手法の確立
6. （バグの無いソフトウェアは存在しないため）バグを修正するための手法の確立
7. 適切なセキュリティが実装されたサーバを用いてサービスを展開する場合の制御方法の確立
8. （もしかしたらSubversionやMercurialの場合もありますが）Gitによるソース管理アプリケーションの動作を保証するための、基本的なセキュリティの実装およびテストの実行
9. ビジネスに則したプロジェクトやコードの記述方法についての適切なガバナンスと相互理解の実装

▶2.1.6 機械学習の採用基準の確立

すでに前項を読んだ読者であれば、機械学習を極めていくうえで必要な知識を持っているかどうか確信が持てなくなっているかもしれません。本書の説明を通じて、読者が自信を持って進めていただけるようになることを願っています。

機械学習の理解を深めるためには、機械学習などのインテリジェントな分析機能を実装したアプリケーションを正しく構築する知識も必要となります。ここではそれらの理解につながる道筋についても示していきます。さらに、社内で機械学習を採用するための正しいロードマップを構築できるようになるために必要な情報も提供していきます。

前項で、機械学習を採用するにあたって必要となるチーム構成とインフラ準備について、説明しました。ただし、どの問題に取り組むべきかについては明らかにしていませんでした。各チームでは解決すべきいくつもの問題があると思います

が、どこから始めればよいかわからない場合もあるでしょう。

インテリジェントなアプリケーションを構築するために必要なチーム構成やインフラ整備がなされていることに確信できている場合、次のステップに進むことができます。次のステップは、解決に値する課題を正しく見つけていくことです。そのためには、以下に説明する一連のステップを進めていきます。設定された課題に対して、最初の段階でいきなり機械学習を導入しようとすることはお勧めしません。チームで取り組むべき課題が機械学習に適したものかどうかを検証するために、必要となるいくつかの重要なコンセプトについて、クローズアップしてみましょう。

最初に、機械学習の課題として適切ではない一連の基準について説明します。

◉――機械学習の利用が不適切な課題定義の基準

1. 人が記述可能な規則やパターンをいくつか組み合わせることで説明できる課題
2. ノイズが含まれたデータが流入する問題があるため、学習可能なパターンを見つけることができない課題（機械学習を実行する際、最大の問題はデータが良くないことです）
3. “利用可能な”データが十分にない課題（一般的に100万個レベル以上のデータ件数があるのが望ましいところですが、課題設定によって必要なデータ件数は異なります）
4. 統計的パターン認識の観点から、1つ以上の入力に対して1つ以上の出力を対応付ける方法を定義できないような課題
5. 100%の精度が求められるような課題
6. シンプルで費用対効果の高い別の方法で課題を解決できるにもかかわらず、特定の技術を使うことを目的としている場合（たとえば、線形代数の数式で読み解けるデータであるにもかかわらず、ディープラーニングを用いることを目的にしてしまうような場合）
7. チームが現在どのようにしてデータを取得し、どのように検証されているのか

理解できていない場合（データの正当性を正しく検証できないため、機械学習を実行した結果が正しいものかどうか判断がつきません）

上で示したものは、機械学習をアプリケーションに実装しようとする際に起こる、かなり一般的な落とし穴です。それらの過ちは、本来の課題解決とは異なる誤った理由から起こってしまいがちです。マネジメントチームは、ビジネスを成功に導くためにインテリジェントなアプリケーションを本当に構築する必要性があるかについて、定期的にステークホルダーなどにヒヤリングを実施すべきです。正しい方法で行えば機械学習は差別化のポイントになりますが、誤ってしまうと時間の無駄になり、長期間にわたりリスクを背負うことになります。特にチームがまだ機械学習の導入経験が浅い場合には、機械学習を正しく実行するために、軽めの事前リサーチや開発に初期コストがかかるので、その点も考慮する必要があります。

もう1つ別の落とし穴は、開発チームのレベルの違いでも起こりえます。チームの中では、チームが得意なある特定の分析手法をレジュメ化し、すべての課題に対してそのレジュメを当てはめてしまう人たちがいます。このような開発手法を我々はレジュメ主導の開発と呼んでいます。レジュメ主導の開発を行ってしまうと、「事業部門にとって有益となる課題解決」という本来の目的を見失い、特定のスキルセットを学ぶことが主眼になってしまいがちです。

◉――機械学習の利用が適切な課題定義の基準

1. 一般的に利用可能な状態になっている大量のデータが存在すること。データの条件をもう少し厳しく設定すると、高品質なデータでデータの偏り（バイアス）がかかっておらず（少なくともできるだけバイアスがない状態であり）、予測したいすべてのケースで利用できるものであること
2. 人がロジックで考案できる規則を使って直接コードを記述するのが難しいような複雑な課題であること
3. （たとえば予測を実行するために適切な変数を使って予測問題を設定した

り、分類問題で正しいラベルが付けられているかを確認したりといった）課題の適応範囲を正しく設定できて、かつチーム内で十分に理解できる課題であること

4. 可能ならば、課題の一般的な事項は事前に定義してドキュメントにしておくこと。課題を正しく理解するために、従来の技術を活用して設計を進めていくのが正しいやり方です。取り組むべき課題が、既存の課題で当てはめることができない場合には、通常、研究者として対応する必要があります。ただしその課題に固有のアプローチでなければ解決できない問題は非常にまれです。一般的には、既存の課題に少し修正を加えることで課題に対応できるようになります。このような方法は、取り組むべき課題に対してビジネスを規定したり初心者が課題を扱ったりする場合の正しいアプローチになります
5. モデルの使われ方が十分に理解されている課題であること。モデルがどのように使用され、誰がどれくらいの規模で扱うかなどを理解することです。モデルを構築する前にできる限り推定してください
6. 入出力の観点で明確な定義が行われ、十分に細分化された課題であること。1つのモデルで多くの課題を混在させて実行してしまうと、正しいパターンの学習を行うことができません。できる限りシンプルなものにすることが推奨されています
7. 因果関係の有無について十分に理解される必要がある課題であるか否かを知ること。特定の産業では、意思決定を行う際に、非常に厳しい規制に直面するケースがあります。規制をクリアできるのはある特定のモデルのみで、そのモデルの使用だけが許されるといった状況が起こりえます。因果関係がしっかり理解できる課題であれば、事前に実装を想定しているモデルにとって課題の適用範囲として問題がないかを確かめておいてください

▶2.1.7 課題の設定指針

多くの場合、自社で機械学習を使い始めようとする際に、しばしば、どこから手

を付けたらいいのかわからないという状況に陥ります。これらの問題は、技術そのものではなく、対応するエンジニアが持っているスキルなどに起因することが多いものです。このような状況では、大きな組織において、実際に機械学習を開始するのが妥当であるかを正しく判断するのが非常に難しい場合があります。妥当性を正しく判断するための成功の秘訣は、自ら試作してみることから始めることです。まず、KaggleのようなWebサイトで機械学習を十分に試すことです。これらで十分な経験を積んだ後は、自社のプロジェクトのうち、小規模で明確な成功を得られるようなプロトタイプを作成することをお勧めします。上記の課題設定のアプローチのいくつかを使って、解決したい課題を選択できる場合には、ある程度の成功を収めやすいでしょう。常にビジネスでの成功を念頭に課題を組み立てるようにしてください。これを無視してしまうと、実験的に機械学習に取り組むという段階から脱却して、商用で使うレベルの課題に取り組むことが非常に難しくなります。

ビジネスの成功のイメージを膨らませるやり方としては、スタートアップや製品管理の観点で考えてみるのが1つの方法です。

たとえば、最初に実装するサービスのプレスリリースを書く場合を仮定します。その際、このモデルが構築されたことで、組織にとってどのような成功がもたらされるかを想像してみてください。“モデルをAIで構築した”というような次元のメッセージではなく、このモデルをサービス展開できたことにより、どのようなビジネスインパクトがもたらされるかについて考えてみてください。明確なリスク／報酬の方程式のような指標を持っておくことで、未開発領域の機械学習プロジェクトに対しても最初の投資に対しての妥当性を説明できます。

また、もう1つ別の視点でビジネスの成功イメージとして考慮すべき項目は、ステークホルダーがプロジェクトの遂行を承認する主な要因がどこにあるのかを正しく理解しておくことです。一般的に、他の課題をイメージしていく場合と同じように、ステークホルダーが抱えている課題について考える場合、あなたにとっての“顧客”は誰で、それらの顧客とステークホルダーがどのような関係を構築してい

るのかを把握する必要があります。人々はそれぞれの立場の違いにより異なる課題を持っていますが、ビジネスとしてお金と時間をかけて解決するのに値する課題は、その中のほんの一部にすぎません。そのため、組織にとって本当に重大な課題を見つけるように努力してください。課題の価値を定量的に計測可能であればあるほどよいです。エンジニアリングチームは多くの場合、他のチームや部門の人たちの価値基準に詳しくないので、それらを計測するのは難しいはずです。そのため、他部門にわたる価値基準を策定する場合には、直接プロジェクトのマネージャに方向性を示してもらうように助けを求めましょう。

2.2 探索的データ分析

前節までの目的は、機械学習に適している課題を見つけて、適切なデータや課題の範囲を特定するための適切な手順を実行できるようになることでした。次のステップは、**探索的データ分析**（**EDA**：Exploratory Data Analysis）と呼ばれる、データの中身を詳しく見ていくフェーズです。取得したデータは一見正しそうに思えることがしばしばあります。ただ実際にデータを詳しく調べていくと、誤りや欠損値などが含まれていることがあり、これらは機械学習で取り込む前に排除されている必要があります。正しくデータの理解を深めていくには、基本的な統計分析やデータ可視化を適用していきます。本節では、探索的データ分析のやり方について説明します。このステップは、扱うデータの種類の違いによって手法が異なるので、読者が参考にされる際には、ご自身が扱うデータの種類を明確に意識しておいてください。

データの傾向がどのようなものかを理解しようとする場合、一般的にアクセスできるデータにどのような種類のものがあるかを理解するところから始める必要があります。たとえば、コンピュータビジョンの課題を調べる場合と、（CSVデータのような）カラム型データを扱う場合ではやり方がまったく異なります。

▶2.2.1 探索的データ分析の直感的理解

探索的データ分析の目的は、解決しようとしている課題に対して有効だと思われるデータの傾向を理解することにあります。データの理解を深めるには、可視化の手法を用いてデータを調べることが重要です。データの特徴について理解していくには、通常、データの種類に応じて異なる方法でデータを操作または変換を行うことになります。この操作／変換とは、自然言語処理（NLP）ではテキストを標準化することで、画像処理では画像を（白黒のように）すべて1つの型の色に変換することであり、さまざまな手法があります。また、CSVでは、いくつかのデータの正規化の形式を当てはめる場合があります。

AI処理フローの中で、図2-3の破線の枠内がEDAフローのフェーズになります。

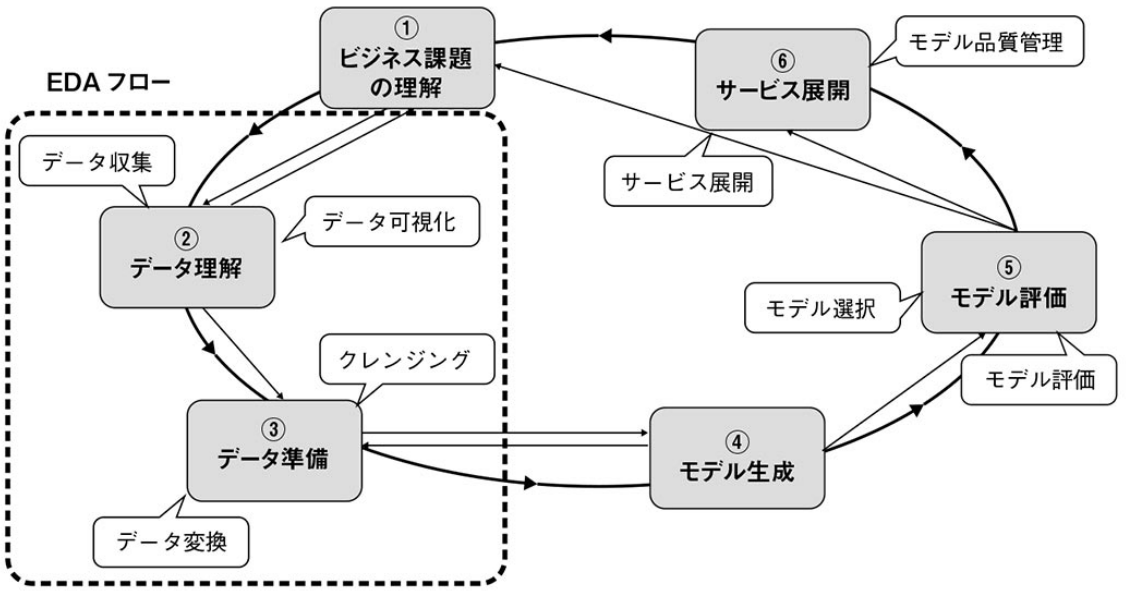


図2-3:探索的データ分析（EDA）のワークフロー

▶2.2.2 データの理解

データの理解にはさまざまな達成目標があります。主なものを以下に示します。

1. 外れ値の特定： 外れ値とは他のデータと大きく外れた値のことで、誤ったデータの一種です。データに外れ値が含まれているかを理解します。外れ

値があった場合には、それらの外れ値が正常なものなのか、実際には誤ったデータなのかを特定することが重要です。これにより、たとえば、データの取得処理やその処理の間で生じるヒューマンエラーを特定して修正できます。

2. データ品質：たとえば、画像の場合なら、物体や事象を認識するのに十分な解像度があるかを確認します。CSVの場合なら、データのさまざまな個所に何らかの傾向が見られることを確認します。
3. 分散（Variance）の取得と理解： 分散とはデータの散らばり具合のことで、扱うサンプルデータの種類によって異なります。画像の場合には、たとえば、画像に含まれる色の散らばり具合を理解することであったり、ある物体が相対的に取り得る位置のばらつきだったりします。また、カラム型のデータの場合には、あるカラムの数値のばらつき具合を調べることなどを意味します。
4. 重要な属性の発見：（分散や主成分分析［PCA］を使って）どの属性がデータの変化に最も影響を与えるのかをランク付けすることもできます。これは機械学習のモデルに影響を与えない属性があるのかを確かめるのに役立ちます。本来、モデルに影響を与えない属性が含まれている場合には、データが無駄に冗長なものとなってしまうため、不要な属性を削除することでモデルの性能も向上します。モデルの精度のチューニング作業においては、不安定性やノイズをあらかじめ減らすことが重要です。画像の場合であれば、検出したいパターンが存在しない画像を削除したり、テキストの場合であれば、文意を正確にとらえるのに不要な文の一部を削除したりすることを意味します。
5. 解決可能な課題か否かの特定： これは多くの人が見落としがちな問題です。データ分析の全体では、複数回にわたってこのチェックを行いますが、このような実装を行うための事前の知識として、機械学習を使って解決できない課題も存在することを想定しておいてください。
6. 意思決定を行うのに十分なデータを持っているかの理解： ある属性を調べるのに必要なデータ量に達していない場合、目的変数（ターゲット）の属性を期待どおりにとらえることができないことに気づくことになります。これはデー

タを取得するプロセスを変更する必要があるかを知るのに役立ちます。データ量が十分か否かについては、データがどのように分布しているのかを見ることで特定することができます。もし1つの属性に分布が集中していて平均的に分散していない場合、データのばらつきのバランスが崩れたデータセットになります。1つのラベルのデータのみが多数存在し、他のものは少ない場合に起きる問題です。データセットを詳しく調べる場合には、属性と目的変数の両方を常に分析すべきです。

7. 課題を解決する方法の特定：EDA（探索的データ分析）を行う場合に、教師ありの手法であれ、教師なしの手法であれ、課題を解決するのに用いるべき方法を見つけるまでに時間を要することがしばしばあります。EDAを実施し、現在取得できているデータを用いて解決可能な課題をできるだけ明確化しておくことが重要です。EDAを用いて課題設定がより細部にわたって理解できるようになることで、解決手法の選択のブレが少なくなり、解決手法の特定が容易になります。

▶2.2.3 EDAプロセスの全体像

これからEDAプロセスの全体像を知るうえで必要な情報を説明していきます。最初に各プロセスで用いられるツールを説明し、その後でEDAの実行方法および正しく実行するうえで知っておくべき内容について説明します。

▶2.2.3.1 EDAプロセスで用いられるツール

◉――クラスタリング

クラスタリングは、K-meansや階層型クラスタリングなどのアルゴリズムを実行することで得られるデータの分類方法であり、データセットの中から同じ特徴を持つ集団を分割する、もしくはそうした集団を摘出することを目的としたツールです。たとえば、クラスタリングを用いてデータの異常な傾向を調べる場合には、データの中でどれが正常のグループに属するかを調べることで実現できます。クラスタリ

ングを使うもう1つの例は、サンプルデータがどのグループに区別されるのかを知ろうとする場合です。たとえば、教師なしデータに対して適切な分類ラベルを付けるというような課題を考えます。その場合、クラスタリングのアルゴリズムを実行することで、データを複数のラベルに分類し、同じクラスに属するサンプルデータの特徴を調べていくことにより、それぞれのクラスに付与すべき適切なラベルを調べることができます。

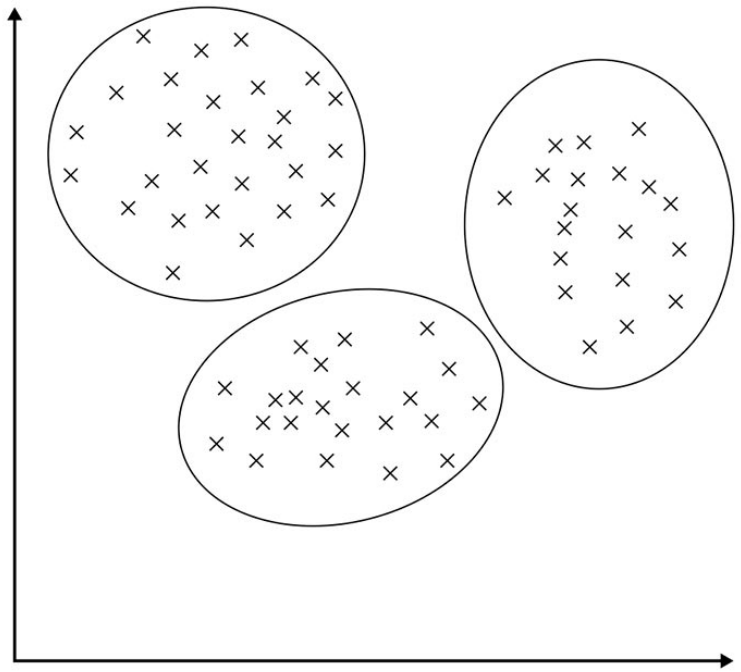


図2-4：クラスタ化されたサンプルデータ

◉── PCA（主成分分析）

PCAは、データの分散が最大となる軸、つまりデータのばらつきが一番大きいところを発見するアルゴリズムです。ばらつきが大きいということは、1つ1つのデータの間の区別がしやすいということを意味します。逆に言うと、データの分散が小さいということは、それぞれのデータ間で重なる部分が多くなる、つまり共通するパターンを持つことになるため、データの特性として区別するに値しないパターンということになります。PCAを使用してデータに最も影響を与える属性が何であるかを可視化してみると、どの属性の関連性が強いか、あるいは低いか、無駄な属

性は何かを理解しやすくなります。属性の分散が大きければ大きいほど、機械学習アルゴリズムによって行われる判断に与える影響の度合いが大きくなることを意味します。図2-5を見ると、2次元のデータの分散が一番大きくなる軸が第1主成分軸になることが理解できます。その次に分散が大きくなる軸は第2主成分軸になります。

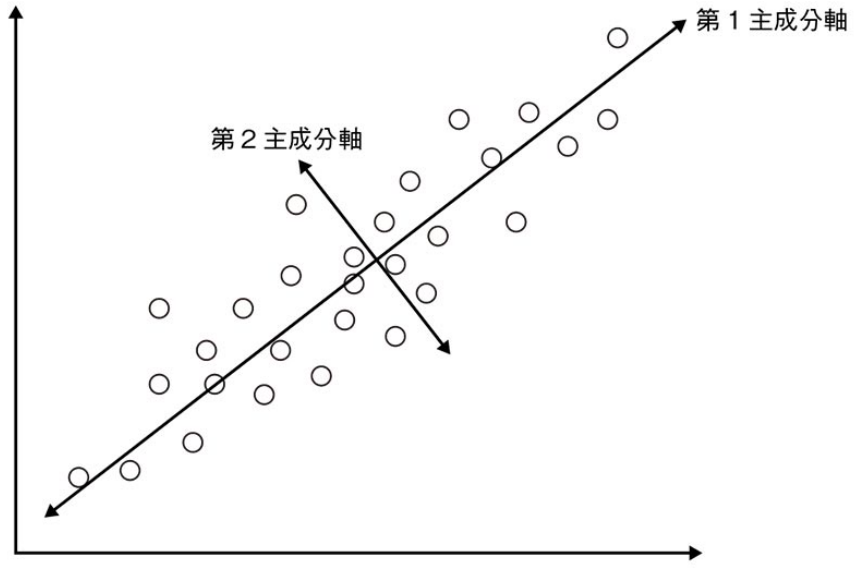


図2-5:PCAの分散の説明

◉――分布の可視化（ヒストグラムや散布図など）

サンプルデータの分布を可視化することは、どの属性の値がどのような範囲に存在するのかを理解するのに役立ちます。データの分布は、解くべき課題の多くの部分に影響を与えます。たとえば、外れ値を調べる（図2-6）、あるいはある属性で正常とみなされる値の範囲を調べる（図2-7）、といったことが簡単にできます。

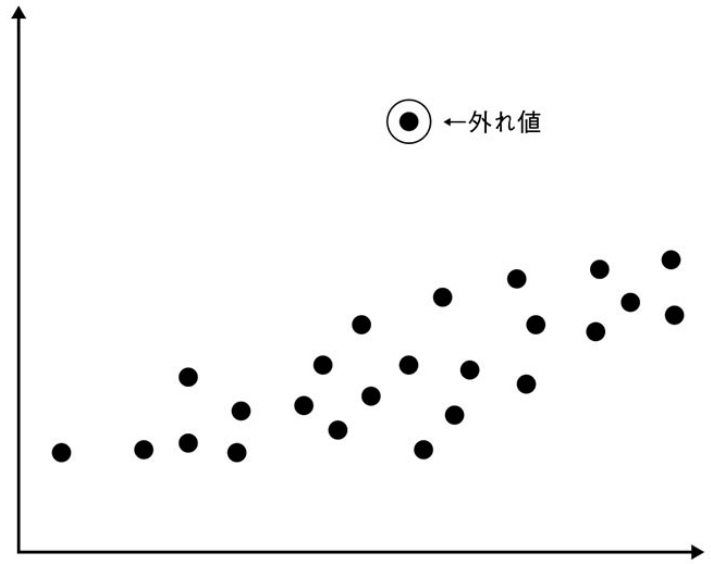


図2-6：散布図の例（外れ値の特定）

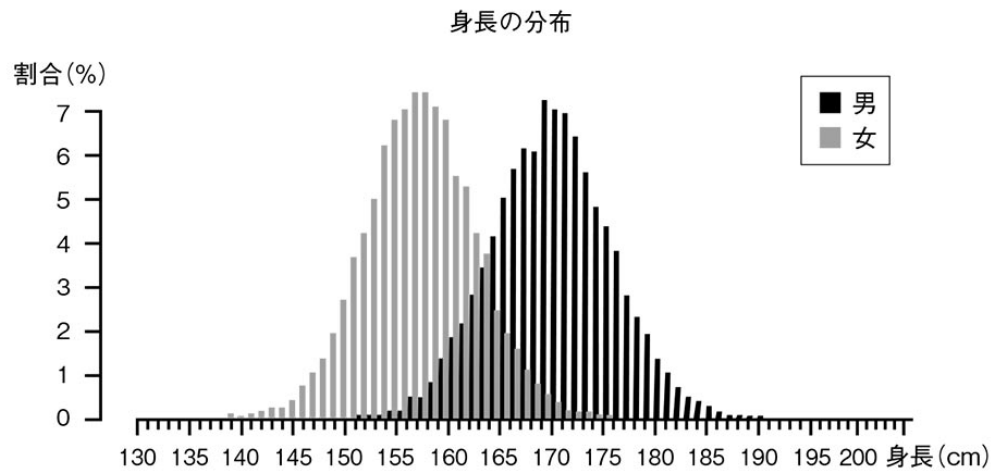


図2-7：人の身長のヒストグラム

◉――ワードクラウド（Word Cloud）

ワードクラウドは、自然言語処理（NLP）のデータセットで最も頻出する単語が何なのかを視覚的に理解するのに役立つアルゴリズムです。以下の例はホテルのレビューを分析した際に、最頻出単語がどのようなものであるかを視覚化したものです。図2-8を見ていただければわかるように、room、hotel、location、staff、breakfastなどが頻出単語であることが理解できます。

図2-8:ワードクラウド―［出典］MonkeyLearn Blog（https://monkeylearn.com/blog/word-cloud-generator/）

◉――ランドマーク（画像認識の精度を上げる技術）

ランドマーク（物体の境界線）は、コンピュータビジョンのアルゴリズムの中で、物体に対する認識の精度を上げるのに役立ちます。

ランドマークを調べるのにさまざまなアルゴリズムがありますが、ここでは近年よく利用されている**SIFT**（Scale-invariant feature transform）について説明します。

SIFTは、一般に画像の中の物体の特徴など**キーポイント**と呼ばれる特徴的な点を検出するのに用いられるアルゴリズムです。キーポイントとして検出される特徴には、物体の端や角（コーナー）、輝度の勾配などがあります。

SIFTは、画像のマッチング、物体検出、シーンの検出など、さまざまなコンピュータビジョンのアプリケーションで使用できます。また、画像のサイズが異なっていたり物体が回転していたりするような状況でも正しく検知できるため、画像分析において汎用性が高い仕組みです。

具体的なSIFTを用いたキーポイントの検出例を図2-9に示します。図2-9の右を見てわかるように、家の窓などを中心に家の特徴的なポイントが検出されてい

るのがわかります。

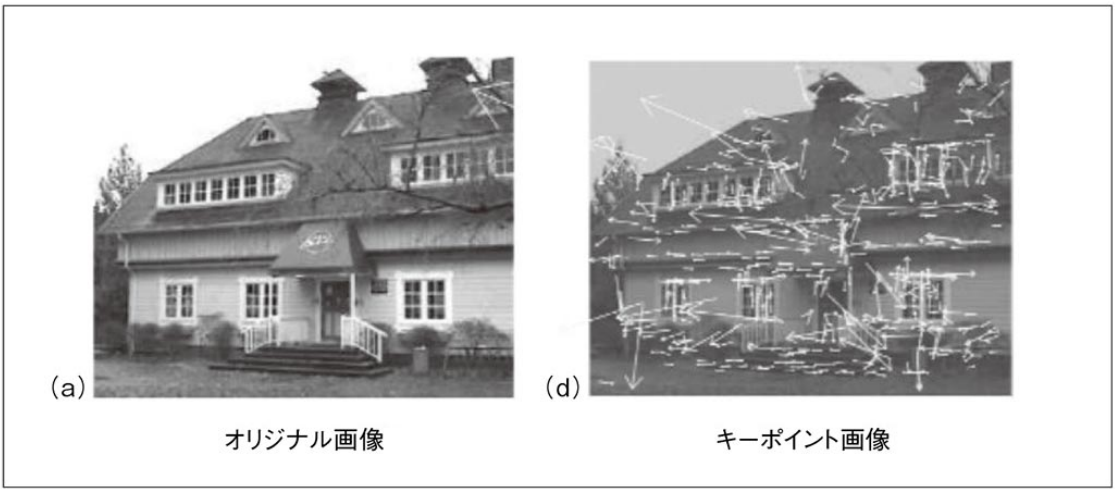


図2-9:SIFT特徴抽出例―［出典］"Distinctive Image Features from Scale-Invariant Keypoints", 著者: David G. Lowe（https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf）

◉――ラベル分布学習（LDL：Label Distribution Learning）

ラベル分布学習は、サンプルデータに含まれるラベルの相対的な重要性を学習するアルゴリズムで、ラベルの曖昧さの問題を解決するために使用されます。

たとえば、図2-10の左の女性の画像を見た場合、正確に何歳かを予測することは不可能ですが、だいたい何歳かの予測はできます。彼女の年齢の確率分布をもって説明をしたほうが、ある特定の年齢をラベル付けして予測するより正解な予測が可能になります。

もう1つの例は、図2-10の右の映画のユーザレビューの満足度です。こちらもさまざまなユーザが点数を付けるため、確率分布でラベル付けされているほうがより正解に予測できるようになります。

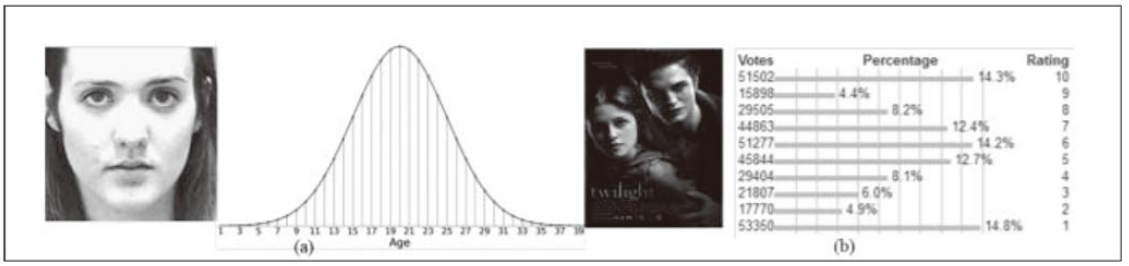


図2-10：ラベル分布学習の例―［出典］"Label Distribution Learning Forests" Wei Shen著　(https://papers.nips.cc/paper/6685-label-distribution-learning-forests.pdf)

◉―― T-SNE/Embedding Visualization（埋め込み可視化）

T-SNE（もしくは**t分布型確率的近傍埋め込み**）は、散布図において高次のデータセットを可視化してどのサンプルデータが互いに類似しているのかを確認することが可能で、PCAで行う次元削減技術と似たものです。この手法は通常、Embedding（埋め込み）など直接的な解釈が不可能な密空間のデータと組み合わせて使用され、どのEmbedding空間が類似しているのかを理解しやすくなります。この一般的な利用例としては、テキストデータの分析で用いられる単語の意味をベクトル形式で表現する**Word2Vec**というアルゴリズムを可視化して理解するケースがあります。

Word2Vecでそれぞれの単語がたとえば1,000個の特徴量を有するEmbeddingの多次元の行列として表現されていると仮定します。Word2Vecで、自身のデータがどのくらい正しくパターン認識をしているかを理解しようとする際には、多次元のデータを散布図などで可視化して把握することが不可能であるため、T-SNEが利用できます。単語が似ているか否かについて、図2-11のような空間距離の遠近（近ければ類似していて、遠ければ似ていない）を用いて表現することにより、視覚的に調べることが可能になります。具体的にはWord2VecのEmbeddingに対してT-SNEを用いて描画しています。Word2VecのEmbeddingにおいてもともと想定される単語同士の意味の類似性が、図2-11のそれぞれの単語の空間的な距離の遠近として正しく表現されていれば、問題ありません。具体的には単語の意味が近い場合には、図2-11で示される単語同士の空間距離が近

く、意味が似ていない場合は空間距離が遠くなります。ただし、似ていると期待される単語同士がWord2Vecを用いたEmbeddingの評価結果と異なって空間距離が離れている場合には、さらにデータセットを拡張するか、異なるチューニング手法を用いてアルゴリズムを再度トレーニングするなどの方法で精度を高める必要があります。

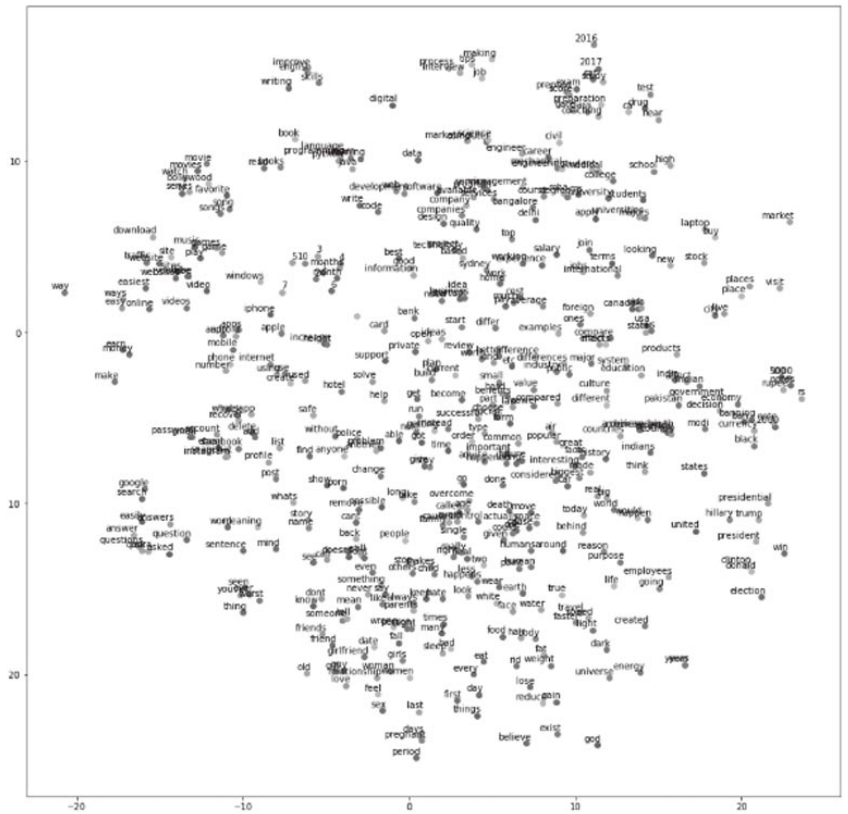


図2-11：T-SNEを用いたWord2Vecの散布図―［出典］Kaggleコンペティションより "Visualizing Word Vectors with t-SNE"（https://www.kaggle.com/jeffd23/visualizing-word-vectors-with-t-sne）

◉――ノートブック

ノートブックは、すばやく実験（experiment）を実行するためのインタラクティブなコーディングを実行する環境としてデータサイエンティストで広く利用されているツールです。ノートブックは、任意のコードを複数行のコードとしてひとまとまりにして実行できるようにする**コードセル**と呼ばれるコンセプトを内包しています。

これにより、ツールの操作に邪魔されることなく、迅速な分析と実験を実行することが可能になります。ノートブックには、コーディングと同じ作業空間で可視化まで実行できる**インラインプロット**という機能もあります。分析のコードが作成（保存）できたら、誰でも同じノートブックを再実行することが可能で、作成した本人のみならず本コードの利用者全員が同様のコードを用いて結果を確認／操作できます。ノートブックにはScala用のZeppelinなどさまざまなものがありますが、代表的なものの1つはPythonで広く用いられているJupyter Notebook（図2-12）です。

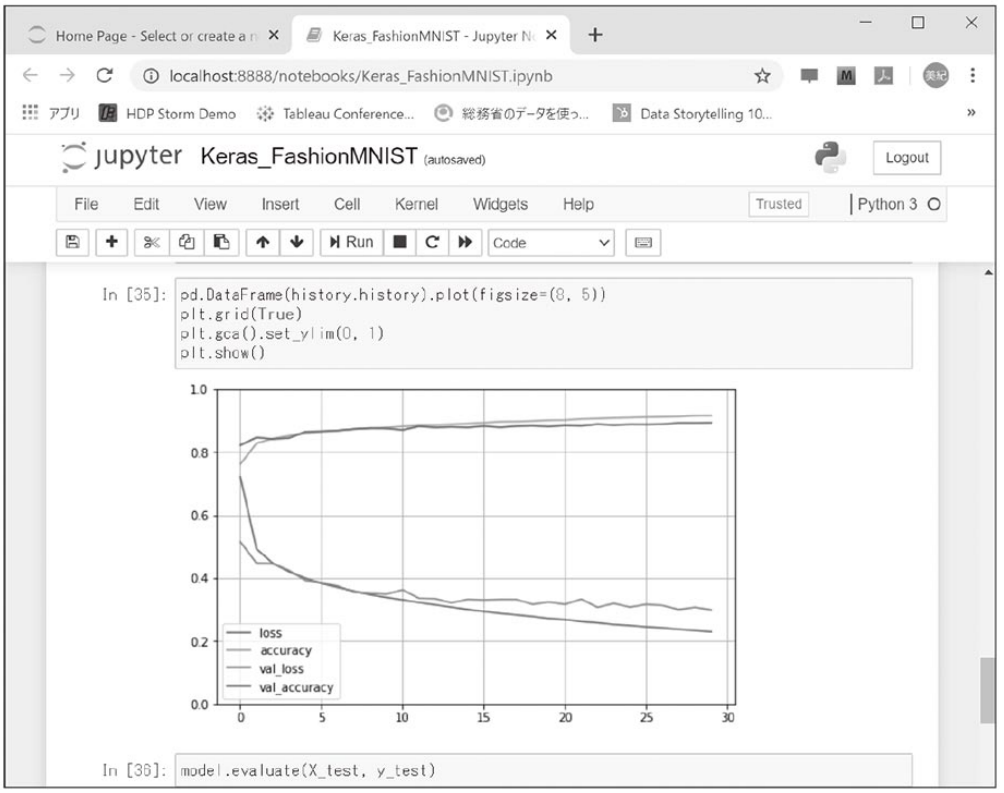


図2-12:Jupyter Notebook上でmatplotlibを用いたグラフ表示例

◉——訓練用 UI

訓練用UIは、モデルを訓練しているときに何が起きているかを表示します。この訓練用UIは一般的に、モデルの変化率、時間経過の中でのテストセットの精度やその他の変化など、データサイエンティストが理解するべき変化を示すものです。チューニングを行う際に、これらのUIで表示されるさまざまな変化を確認す

ることが重要です。それにより、モデルが変更されたり過剰適合したり、不良なモデルや入力のデータセットによって何らかの症状が発生したりしていることを理解できるようになります。

どの訓練用UIを使用するかは、利用するプログラミング環境によって異なります。たとえばTensorFlowを用いる場合には、機械学習の実験に必要な可視化ツールとして、図2-13のようなTensorBoardと呼ばれるものが実装されています。このツールを使うことにより、損失や精度などのトラッキング／可視化や、入力データの表示などが簡単に行えます。

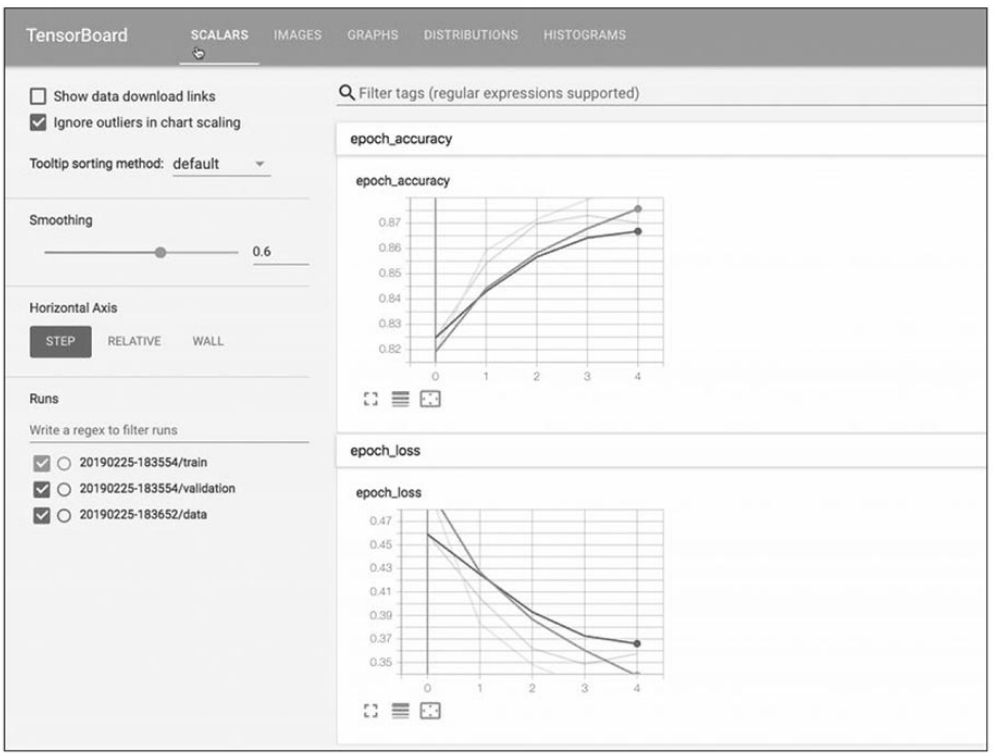


図2-13:TensorBoardの可視化例―［出典］TensorBoardの公式URL（https://www.tensorflow.org/tensorboard）

TensorBoardの機能についてさらに詳しく知りたい方は、図2-13のキャプションに記載したURLのページの情報を参考にしてください。

◉――データ加工／分析変換ライブラリ

データ加工／分析変換ライブラリとしては、Pythonのプログラミング言語で記述されたデータ加工／分析用の拡張モジュールのpandasをイメージしていただけるとわかりやすいです。これらのライブラリは、機械学習のアルゴリズムで使用するデータを変換するための操作に関する基本的なルーチンが含まれています。

◉――数値演算ライブラリ

数値演算ライブラリは、プログラミング言語のPythonで記述された数値演算を効率的に行うための拡張モジュールNumPyをイメージしていただけるとわかりやすいです。数値演算では、通常、配列やテンソルを扱えるようにするために、ndarray（N次元配列）もしくはテンソルの抽象化の実装が含まれています。利用頻度の高いさまざまな数学の演算が実装されています。

これらのルーチンのコアはCやC++で記述されています。その理由は、CやC++がオペレーションシステムなどのカーネル（コアな部分）の記述言語であり、さまざまな種類のハードウェアの最適化を最大限活用できるからです。また、NumPyのライブラリのコアロジックもCやC++で実装されて、Pythonがインターフェース言語になっており、このライブラリをPythonで呼び出して利用することができます。

▶2.2.4 評価指標の使用方法

課題の正当性を評価する場合、目的の違いにより、さまざまな評価指標が存在します。評価指標の選択によって精度の評価が変わるため、課題に合わせて適切な評価指標を用いる必要があります。

限られた紙面ですべてを取り上げることは不可能なので、ここでは2例を示すことで、評価指標の使われ方をイメージできるようにします。

たとえば、モデルで利用されるデータが1つのクラスに偏っている場合、クラスの偏りで誤った評価に陥ることがないように回避しながら高い精度の評価を保つ

必要があります。このような場合の評価指標としては、正確度の代わりに**F1スコア**（F値）を使用することをお勧めします。F1スコアは直接的な正確度と比べて、さまざまなクラスのデータに対してバランス良く評価を行える指標です。正確度を用いた評価では、バランスの取れた精度が正しく評価できず、データセット内の1つのクラスに高得点を与えるような評価が行われます。正確度もしくはF1スコアのどちらを使うのが正解であるかについては、解決すべき課題によって異なります。

たとえば、スパムメールを見極める例を考えます。通常、正常メールのほうが圧倒的に数としては多いため、ただ単に正常メールなら正常とラベル付けし、正しく正常と判断する確率を計算する方式である正確度を用いても、迷惑メールの見極めにはまったく意味をなしません。そのため、迷惑メールに対して正しく迷惑メールと判断できた場合に評価スコアが上がるように、評価指標であるF1スコアを用いる必要があります。こちらは一例になりますが、F1スコアを含め具体的な評価指標の代表的なものについては、のちほど説明します。

もう1つの例として、物体認識での評価指標を使用したものについて説明します。物体認識では、場合によっては、画像内に重なる複数の物体を評価する必要があります。その際、シンプルなF1スコアを使用するだけでは正しく評価できません。そのため、物体認識では、一般的に境界ボックス（Bounding box）と呼ばれる仕組みで画像を分割します。境界ボックスには、それぞれ独自のスコアと精度を設定できます。ボックスごとに評価するのは、物体認識の課題を扱う場合にかなり一般的に行われる手法です。この仕組みにより、物体認識モデルの出力に対して、さらに精度の高い精密な分析が可能になります。

このように、取り扱う課題に合わせて、評価指標として適切なものを選択する必要があります。それに加えて、そもそも評価指標を使用するのに前提条件が存在することもあり、評価指標を使用する際には、さまざまな条件をできるだけ事前に把握しておくようにしましょう。

▶2.2.5 EDAの繰り返し実行

上記のEDA（探索的データ分析手法）を繰り返し実行することにより、データを理解し課題を解決するための入力データと、目的変数にあたる出力データセットの一部として探しているターゲットについての理解が深まり、両者の間に何らかの傾向があることの確信が得られるようになります。EDAはデータの理解を深めていくプロセスであるため、一般的には、複数回行ったり、しばしば他のプロセスからEDAに戻ってきたりします。

人々が機械学習のモデルを構築する際に見落としがちなのは、モデルの開発に多くの時間を費やすのに対し、入力データの理解についてはあまり時間をかけていない点です。しばしば我々が実施したいこととして、課題に対して入力データが利用可能なものになっているか、もしくは十分に動かせるかを実感するために、EDAのプロセスの一部で機械学習のモデルを構築して使ってみたくなります。その場合には、モデルのチューニングには極力時間をかけずに、データセットの理解に注力することが重要です。EDAのフェーズではデータの理解に注力することにより、それ以降のプロセスでは、各プロセスの機能の理解に集中することができます。たとえば、モデルの構築フェーズではモデルのチューニングに注力できるようになります。

一般的にEDAのプロセスの一部には、さまざまなモデルのうちどれを使うべきかといったトレードオフを評価することが含まれる場合もあります。また、試せる可能性のあるモデルの種類を特定することも含まれます。

EDAのプロセスに何度も立ち返る理由は、モデルからの出力として最高のスコアを得るために立ち返ることもありますが、そのようなケースよりも、さまざまな仮説をテストするために何度も立ち返ることが多くなっています。入力データの内容やラベルの分布などその他の変数については、常に最初の段階でできる限り精査して扱う必要があります。

一般的に仮説を立てるための方針として役立つのは、以下の内容を検討することです。

1. 課題の種類に応じて、一般的によく用いられる特定のモデルの種類を試すのがベスト
2. 入力データの種類に応じて、一般的によく用いられる特定の変換を行うことで、モデルの性能が向上
3. 入力変数の特定の組み合わせがベストな結果を生み出す
4. 1つの課題に対して1つのアプローチで対応するのがベスト

基本的に、科学的手法を使って考えているのと同じように、EDAにおいても仮説を立てて、1つの変数以外のすべての変数を定数に保ったまま、その仮説をテストします。

1つ以上のモデルのテストを実行してみて、入力データがいずれのモデルに対してもある程度期待される結果が得られる安定したものになっていることが確認できたら、さらにモデルのチューニングに集中します。モデルのチューニングでは、入力データのパイプライン処理、ハイパーパラメータやアルゴリズムの変更などのチューニングも必要になります。

一般的に、初回のEDAではデータを調べ上げることに注力し、データを理解してデータの傾向についての知見をドキュメント化したり可視化したりすることに、全体の時間の約90%を使います。2回目以降では、モデルの開発に向けた知見を得ることが含まれるため、EDAでデータを理解するための反復の時間はより長くなっていきます。モデル開発の初期の段階では、モデルの開発自体は実際にプロセスの重要な部分ではありません。一般的に、最初の段階では、検討に値しない条件やデータを可能な限り排除して、データの理解が深まるにつれて、徐々にモデルの開発に注力していくように進めていくのが正しいアプローチです。

実際にモデルの訓練を始める前に、EDAのプロセスにおいてデータについてさまざまな観点で複数の可視化を行い、データに含まれる属性の検出や、さまざまな種類のノイズが含まれたデータの削除を行っていきます。

モデルの開発を進めていく中で、通常は訓練のプロセスの一部としてEDAを

組み込み、モデルのアルゴリズムに取り込むのに適した安定したものとして入力データを変換する必要があります。そのために、メモを取りながらデータに必要なさまざまな変換を見つけ出していきます。一般的に、機械学習のモデルの入力に適した行列を作成する前に、入力データが適切なものか、課題に適したモデルが構築可能であるかについて方向性を見極める必要があります。その場合、よりシンプルなモデルを使って訓練を実行するのに適した状態にもっていきます。そのために、入力データに対していくつかのデータ変換を行う必要があります。

一連の経験処理（訓練プロセス）については、パイプラインで処理すると理解しやすくなります。ここで言うところのパイプラインは、入力データの変換から精度の出力までに必要なそれぞれの処理ステップのワークフローとして定義されます。パイプライン処理の最後のステップには、精度スコアの検証が含まれます。精度スコアの検証を通じて、入力データとモデルの性能の両方が良いものであるかを確認することができます。

このパイプラインの形式で検証を行う理由は、処理品質を向上させやすくなるからです。最終的に商用での利用に耐えうる品質のモデルが出来上がるまでには非常に多くのプロセスを経ることになります。パイプラインの一部を段階的な検証のパイプラインとして組み込むことにより、データやモデル選択の正当性などをそれぞれの段階で確認することができるようになるのです。

パイプラインは一般的に以下のステップで構成されます。

1. 入力データの変換
2. 入力データ変換後のデータを処理するための1つ以上のモデルの構築
3. モデルを用いた精度の出力

商用でのサービス展開時点での注意点になりますが、最終的な検証が終わった段階では、上記の1番目と2番目のステップの結果を再利用することが多くなります。長年にわたり商用利用を続けていくにつれて生じる性能劣化などをフィー

ドバックするためのループを構築するために、出来上がったモデルの精度を定期的にモニタリングする必要があります。一般的なフィードバックループでは、推論のパイプラインについては通常、最初の2つのステップのみを実行していきます。ただし、長年利用していくことで生じる可能性があるデータやモデルの劣化をモニタリングするために、定期的に3番目のステップも組み込んでください。

具体的なモデル開発フローは、図2-14の破線部分となります。

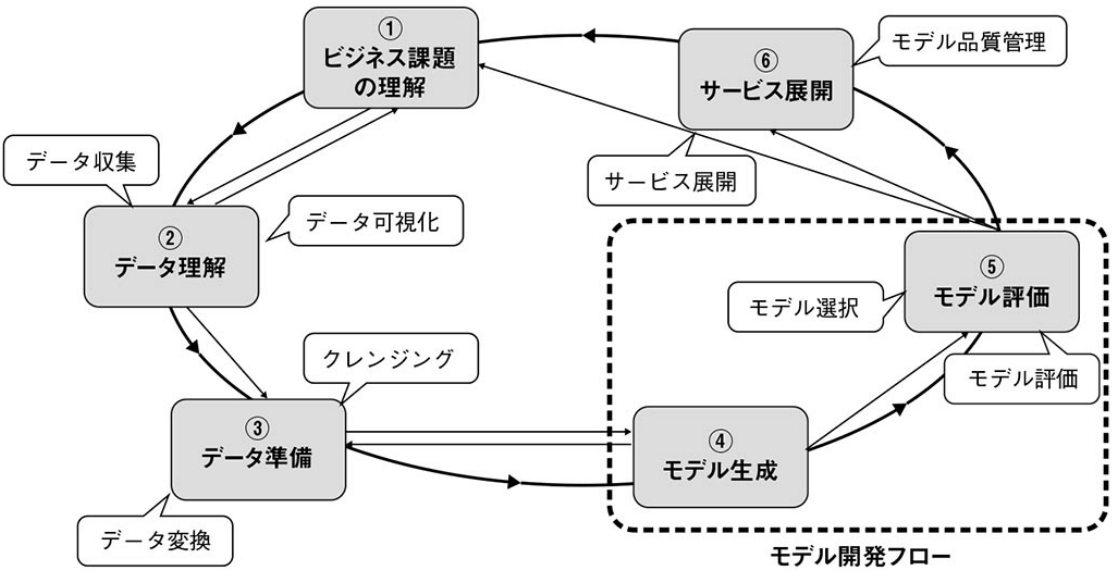


図2-14:モデル開発フロー

▶2.2.6 パイプラインの不具合の調査方法

機械学習で使用するデータセットを精査するときには、潜在的な問題を考える必要があります。それらの問題を診断するのに役立ついくつかのツールはのちほど紹介します。具体的なツールを説明する前に、潜在的な問題とはどのようなものなのかを説明します。

以下に、データセットを精査するときに共通する問題を列挙します。今までの説明の中にもいくつかのヒントが含まれていますが、ここでさらに詳しく説明します。

1. バランスが崩れたデータセット

   ●1つのクラスのデータは多量に存在するが、他のクラスのデータ量は少ないといったデータセットを用いる場合。特に、不正検知のデータセットなどが好例です。

2. 不十分なデータ量

   ●特に、与えられた課題に対して、少数のサンプルデータしか存在しない場合、実世界の特徴を表現しきるだけのデータ量に達していないときがあります。この場合、機械学習で必要となる特徴を表現することができません。機械学習では、実行されるデータタイプに対して十分なサンプル数が必要です。

3. ノイズの多いデータ

   ●外れ値や欠損値などの誤った値などがデータに含まれている場合、分析に利用できるデータに至っていません。

4. 分離不可能なデータ

   ●2つのサンプルデータの間の違いを明確に示すことが難しい場合です。たとえば、データの属性の値はほとんど同じですが、サンプルデータとしては2つあり、別個の値として存在している場合です。

5. バイアスがかかったデータセット

   ●バランスが取れていないデータの場合と基本的な考え方は同じですが、入力データのサンプルデータの母集団のうち特定のタイプのサンプルデータのみを反映していて、他のサンプルデータの特徴を反映していないようなデータセットになっている場合です。

6. モデルの不適切な用途の設定。特定用途のモデルを他の用途に転用してもうまく動かないようなケース

   ●たとえば言語モデルを考えた場合、英語で学習されたモデルが存在する場合に、そのモデルを使って日本語を学習／推論してもなかなかうまく動かないというようなケース。

   ●もう1つ別の例としては、コンピュータビジョンの場合にニューラルネットワークを白黒の画像で学習させたにもかかわらず、カラーの画像を学習させるような場合。

■ COLUMN　AutoML（自動機械学習）とEDAの関係

AutoMLは、データセット中の特徴量やモデルの選択などのサーチを、力ずくかつ自発的に行って、機械学習を自動構築することであり、比較的新しい概念です。AutoMLは、入力のデータセットの変数として最適なものを生成することに加えて、特徴量の選択やモデルの選択についても自動で行う仕組みを持っています。

一般的に、探索空間（Search Space）については、いまだにユーザが決定する必要があります。これらのライブラリは改良されていますが、通常、著者はAutoMLを使う前に、機械学習の基本をマスターしておくことを推奨しています。その最大の理由は、モデルの管理とチューニングを行う際に人の手による適切な判断が必要になるからです。

たとえば実際の結果が望みどおりになっていないことがよくあります。そのような場合、モデルを自力で修正できる能力や、モデルで選択された探索空間がどのようになっているかを理解しておくことが重要です。

通常、AutoMLで期待された結果が得られない場合にその理由の調査を進めていくには、モデルの訓練時の実行結果を記録した試行のログを調べていくことがキーポイントになります。試行のログを調べていくことは、自動特徴量選択の手法も同様ですが、探索空間を微調整する際にも役立つ場合があります。

AutoMLは、どのようなモデルが良いものであるのかを探索するためのツールとして有用であり、課題やデータセットの理解を深めるのに役立ちます。先述したプロセスのいくつかは自動的に実行されるため、どの変数が有用であるとか、優れたプロトタイプのモデルがどのようなものであるかを調べるために、探索空間を設定して出力内容を確認してください。

▶2.2.7　プロジェクト範囲の設定方法

のちほど詳しく説明するモデル開発における経験（訓練プロセス）の範囲を決定するために、一般的に機械学習のモデルを適用するのに必要な基準について

把握する必要があります。最初に重要なことは、短時間に達成可能な目標を設定することであり、プロジェクトの範囲を最小となるように限定しましょう。広範囲に及ぶ課題を一度に解決しようとすると、クリアすべき課題が複雑になり最後まで正しくやりきることがかなり難しくなるため、好ましくありません。課題を小さな単位に分割し、それぞれを小さな単位の課題とすることで、結果をすばやく確認できるようにしてください。適切なチームを構築してプロジェクトの適切な範囲／領域を理解する仕組みを確立するために、前述の基準を参考にしたり活用したりしてください。

いったんプロジェクトの範囲／領域が理解できると、解決しようとしている最小単位の課題のみに注力して実験を構築できます。そこまでいけば、その最小単位の課題に適したデータやモデルの特別なケースを発見して、訓練用のデータセットと本番環境の間で生じる問題についての理解が深まるため、正しく課題をクリアできるようになります。

プロジェクトがどのように進んでいくかを推定して、最も効果が高いと目される特定のバージョンの経験に最も多くの時間を割くように常に調整してください。また、いつでもプロジェクトを強制終了したり再開したり、もしくは課題に対して異なる手法を使えるように柔軟性をキープしておいてください。さらには、選択した課題のアプローチに固執しすぎないでください。そのためには、すべてのステップで常にその課題のアプローチが正しいかどうかも含めて冷静に評価するように心掛けてください。もしチーム内で与えられた課題に対する評価方法がわからない場合には、経験のさまざまな属性をドキュメント化し、他のプロジェクトのステークホルダーなどと話し合って解決策を見つけてください。

▶2.2.8 プロジェクト期間の見積もり方法

最初の経験を構築する際に、今まで学んだことをすべて把握し想定することはできません。読者の方々は、今まで説明したすべての落とし穴を避けながら機械学習の適用を始める方法を、自問しながら模索していくことになるでしょう。本項

では、この章で提起された懸念を具体的にイメージしていただくために、1つの経験を構築する方法に対するベストプラクティスを説明します。

本項では、設定された目標を達成するために検証済みのデータセットがあり、そのデータに基づいてモデルの構築を開始することを前提としています。このフェーズの前提条件としては、データセットの評価が終わっていて、今までの分析による経験的証拠に基づいて、より合理的な根拠で正当化できる課題を実装して洗練を開始したいと考えている状態になっているものとします。

一般的に、EDAのプロセスを通じて、以下のことを理解します。

1. データの傾向についての仮説
2. データを使って試したい1つ以上のモデルの選択方法
3. モデルを検証するための明確な評価指標

この実験では、以下のような潜在化している他の要件についても特定する必要があります。その要件の中には、試した課題が一般的な課題解決に利用できるものなのかを特定していくことも含まれます。

1. メモリの使用量
2. 商用環境への展開の実現可能性
3. モデルの基礎的なベンチマーク
4. 与えられた課題に対して期待される精度の範囲
5. 汎用的に利用可能なものとしてその課題を採用できるか否か

▶2.2.9 問題の未然防止策

実際のところ、このような状況を初めに期待することはできないのですが、すでに利用すべきデータとしては適切なものが得られています。そこで次にすべきこととして、「課題を解決するために使用すべきモデルの選択」と「それらをどのよう

に検証するのか」について理解する方法を知るべき段階にあります。これらを正しく実行するには、実験リストを作成し、チームの他のメンバーと話し合い、仮説がチームの目標とするKPI（Key Performance Indicator：**重要業績指標**）に適合していることを確認してください。多くの場合、モデルの構築にあまりにも早く着手してしまう結果、誰のために構築しているのか、なぜモデルを構築しているのかについて思い起こすことがなくなってしまいがちです。以下の手順は、チームの関わりを促進し適切なものを構築するために正しい方向性を維持するのに役立ちます。

▶2.2.9.1 EDA 結果の検証手順

EDAの結果を検証する手順は以下のとおりです。

1. 最初に、要約されたレポートから調査結果を収集します。アプリケーションが商用環境で稼働し、維持することの承認を得るために、ステークホルダーに対してレポートを共有できていることを確認してください。データサイエンティストのチームが顧客である場合は例外になりますが、絶対にアイデアに基づいたプロトタイプの作成までを目標とせず、その後、商用でどう稼働されるべきか、というところまで必ず考えてください。そしてさらに、我々は企業のインフラストラクチャ担当のチームに相談して、モデルを適切にサービス展開する方法を理解しておくことを強くお勧めします。インフラストラクチャ担当は、IT部門かクラウドのアカウントを管理する人のことを指します。
2. それぞれのステークホルダーに対して調査結果を理解する機会が得られたら、できる限り実験を減らしましょう。すべてのアイデアについて、実験を実践しながら利用価値を追求することは膨大な時間を要するために避けるべきです。実験を行う際には、アジャイルスタイルのアプローチに注力して、結果を見た人からすぐにフィードバックを得られるようにしましょう。ビジネスのステークホルダーとエンジニアリングチームには製品化の締め切り期限があり

ます。彼らはあなたが開発したモデルの出力を使用するために何が必要かを理解する必要があり、またこの機能をいつ出荷できるかについても知る必要があります。これらのことについてお互い合意し、期限どおりにあなたがモデルの開発を完了すれば、あなたが開発したモデルが採用されるわけです。おめでとうございます。これであなたもソフトウェア開発のライスサイクルの一員に正式に加わることになったわけです。

3. 2番目の内容を繰り返し実施し、あなた自身がチームのルーチンに定常的に組み込まれることで、モデルのサービス展開をより早くできるようになるだけでなく、どのモデルがこの環境で受け入れられるかについてさらに正確に理解できるようになります。
4. この段階になれば、チームに成果を示すために、どのような指標をステークホルダーなどが見たがっているかについての情報を収集しましょう。ステークホルダーによって、見たい指標は異なるはずです。パワーポイントやノートブックを使って表にして、試しているモデルのトレードオフを明確に比較検討できるようにしましょう。現在試しているモデルの検証をさらに進めていくと、最終的にはモデルの調整の限界を知ることになります。検証結果が十分に期待どおりの目標をクリアしている場合は問題ありませんが、期待どおりの成果が得られない場合、EDAによる調査に立ち返る必要があります。
5. 商用でのサービス利用と利用すべきモデルが1つ決まると、次は、モデルのメンテナンス計画を検討します。商用環境でモニタリングしなければならない要件を見つけ出しましょう。運用を継続していくうちに何度かモデルを再検討する必要があるかもしれません。チームと協力してメンテナンスの業務範囲を明確に定めます。のちほどさらに詳しく触れますが、モデルの維持とメンテナンスは、複数の分野にわたるタスクです。ここではモデルの構築と機械学習に関するチームのサポートに重点を置く必要があります。通常、すべてのメンテナンス項目について詳細にわたって自分で処理する必要はありません。機械学習を使って成功を収めるには、さまざまなグループ連携が必要となるた

め、コミュニケーションが最も重要な事項となります。

2.3 モデル開発の下準備

今までのところでデータの理解からEDAまで説明しました。ここまで準備できればモデルの開発に入ることも可能ですが、さらに効率的にプロジェクトを回していくために、以下の項目について検討することをお勧めします。

▶2.3.1 データ準備の最適化

基本的なデータ準備の処理プロセスであるデータパイプラインが整っていると仮定した場合、ほかに何ができるでしょうか。最初に行えることは、ベクトルを生成する方法も含めてデータパイプラインを少しずつチューニングし最適化していくことです。以下にそれらを実装するためのヒントについて説明します。以下の内容によりすべてが網羅されているわけではないので注意してください。

1. データパイプラインで再利用可能なものはできる限り利用してください。出力が共通して利用可能なものでしたら、キャッシュに置いて高速に使えるようにしてください。たとえば、ベクトルを毎回同じ方法で作成している場合、データの読み書きをする際には、データがローカルなデバイスにあるか否かによらず、利用可能なものがすでに存在していないかチェックする機能を追加してください。そうすることで、データパイプラインを構築する際に大幅な時間短縮が図れるようになります。
2. 疑似乱数を生成する場合には、基本的に毎回同じ疑似乱数を生成するようにシードを設定してください。一般的に、疑似乱数では毎回異なる数値が生成されますが、分析で使用するデータセットを生成する際に、疑似乱数が毎回同じになるようにするために、シードを利用してください。そうすることで、毎回同じ結果を得ることが可能となるので、分析がしやすくなります。

3. データパイプラインの中で明らかなボトルネックを見つけて取り除いてください。たとえばプロファイリングについては気にしすぎないようにしてください。これはプロジェクト全体を長期的に時間短縮するのに役立ちます。データパイプラインは頻繁に再利用することになります。データセットの生成にかかる待ち時間とモデルの訓練の時間を短縮することで、全体の開発時間の短縮に役立ちます。
4. 可能な限りデータパイプラインのコードをモジュール化してください。モジュール化されているほどのちのち再利用がしやすくなります。
5. データパイプラインに十分なログ機能を追加してください。パイプラインのどこで障害が発生したかがわからないときに、ログはパイプラインの構築中のどの段階にいるのかを知る手掛かりになります。
6. ベクトル化されたデータを蓄積する際に、使用するデータの型に気をつけてください。完全な64ビットの精度が必要ない場合、32ビットの精度を使うことでメモリ容量を抑えることができます。一般的に32ビットの浮動小数点数で十分です。64ビットの浮動小数点数は多量なメモリを費やしますが、ディープラーニングでは多くの場合、必要ありません。
7. データが蓄積されているデバイスに注目してください。GPUが不要な場合には、使用しないでください。CPU上のデータパイプライン用に生成された多くのデータは、できるだけそのままの状態をキープしてください。ニューラルネットワークの内部で使用されないデータは、特にGPUのメモリを無駄遣いするだけなので利用しないように注意してください。
8. データセットを出力する場合には、事前に訓練・テスト・検証用にデータを分割しておいてください。

▶2.3.2 モデルのアーキテクチャの最適化

以下に、モデルのアーキテクチャを最適化するためのヒントを説明します。こちらもすべては網羅していないため、その点はあらかじめ注意してください。

1. 最初はシンプルなモデルの訓練から始めてください。初めに繰り返し実行すべきモデルの優先度を決めてください。
2. モデルの訓練や推論のプロセスからデータパイプラインを切り離してください。複数のモデルのアーキテクチャに対して同じデータパイプラインを使用するかもしれません。1つの仮説について、1つのデータパイプラインに対して複数のモデルを組み合わせて用いる場合が一般的なので、複数のモデルで同じデータを試すことを前提に実装してください。
3. データパイプラインと評価指標を使用する場合も、「データの型」と「モデルが実装されているデバイスの仕様が十分であるか」に注意してください。モデルによっては数ギガバイトの容量になったりします。
4. モデルに説明可能性が必要であるか否かに注意してください。説明可能性が必要な場合は、そのためのコードを作成するか、説明可能性を実証するために、事前に使用する関数を把握しておいてください。
5. データを訓練用・テスト用・検証用に分割してください。
6. モデルに使用されるデータに対して、前処理としてデータの修正や正規化などのクレンジングがなされていることを保証してください。
7. 訓練中に現れる不正オペランド（NAN）を検知できる仕組みを実装してください。NANなどの効果によって収束しない可能性があるモデルを扱う場合には、訓練用のUIを使って、訓練が収束に向かわず拡散に向かうタイミングをチェックしてください。ログを記録しておくことも状況を改善するのに役立ちます。
8. チューニングをする際、時間経過に伴う勾配の変化は、最初にモニタリングするべき指標の1つです。

▶2.3.3 データサイエンスのコンペティションの利用について

多くの読者は、優れたモデルを構築するために、データサイエンスのコンペティ

ションのプラットフォームを使用してきていることでしょう。それらのデータセットは前処理が終わっていて、データのクレンジングがなされており、課題の範囲も事前に指定されています。それらのコンペティションでの経験は確かに課題解決を実践するのに役立ちます。しかしながら、それらのモデルや課題設定の目的は、完全な実社会を反映したものではないことに注意してください。

コンペティションのKPIとしては精度を上げることが重要なため、モデルは過剰適合するように構築されます。また、データセットは事前にクレンジングされているので、実際に構築や経験で必要となる大部分の作業を経験する機会が奪われてしまっています。

これらのコンペティションで訓練を積む場合、多様なユースケースでの経験を積むことに注力し、個々のワークフローの最適化にとらわれすぎないでください。

2.4 モデル開発

ここでは、一般的にイメージされる機械学習のプロセスであるモデルの開発の説明に入っていきます。

モデルの開発は、モデルの作成や評価のプロセスで構成されていて、多くの書籍やブログなどでカバーされている内容です。そのため、ここでは、それらの情報ではあまり触れられることのない項目を中心に補足的な説明をしていきます。

▶2.4.1 経験（Experience）の実行

経験（Experience）とは、入力データによって機械学習のモデルが訓練されること、つまり新たなデータに基づきモデルが経験値を上げることを意味します。そのため、機械学習ではモデルの作成からの訓練プロセス全体を経験と呼ぶことがあります。経験のフェーズで注力すべき点は、機械学習のモデルの作成と精度検証をすばやく繰り返すことです。あなたの目標は、十分な分類や予測精度を満たすモデルを構築するのに加え、開発を注力するのに値するモデルがどのような

ものであるかを見極めるのに十分な経験を積んでいくことにあります。そのため、経験のプロセスを何回も実行し、実社会で使用できる1つ以上のモデルが出来上がるまで、実験を微調整する必要があります。

▶2.4.2 経験の実行ステップ

経験を効率良く実行するために、ステップごとに実行を正しい方向に導くのに必要な検討項目を見ていきましょう。経験の実行に際してあらかじめ検討しておくべき内容は以下のようになります。

1. 評価指標の選択のステップ

   良いモデルがどのようなものであるかを評価するための指標を特定します。たとえば、次のようなトレードオフを検討し、同僚に説明できるようにしましょう。二値分類を実行する場合、評価指標としてAUCを選択するでしょう。答えが誤っていた場合のペナルティやリスクは何でしょうか。それらのトレードオフについて理解するようにしましょう。

2. アーキテクチャ選択のステップ

   構築したいアーキテクチャを1つ以上特定しましょう。おそらく、すでに公表されている論文やGitHubのレポジトリから参考になるものを見極め試すところから始めましょう。

3. データ変換のステップ

   生データを機械学習で利用可能なベクトルの形に変換するための適切なデータパイプラインを構築しましょう。このコードは再利用可能で理解しやすいものになるようにしておきましょう。このコードはのちのち別のプロジェクトや同じプロジェクトでも別言語での実装が必要になった場合、再利用もしくは変換し直したりするからです。特に有用なものになれば、大規模なシステムで使用したり、ほかのチームで再利用したりする場合があります。

4. パラメータの設定ステップ

   パイプラインの設定が完了すると、パラメータの基本の組み合わせを設定して、モデルの訓練を開始します。まず小規模のデータセットで過剰適合させて、パイプラインをテストするというループを、すばやく繰り返します。経験を繰り返すことで、1番目のステップで定義した評価指標を使って最も価値あるモデルを見つけます。

5. モデルの再現／複製

   その後、モデルを微調整し、検証結果を再現および複製できることを確認します。

▶2.4.3 評価指標（Metrics：メトリクス）の理解

モデルを評価するために、評価指標の選択は極めて重要です。評価指標はモデルの種類の違いによって大きく異なります。通常、最終的に1つのモデルに絞り込む選択をする際に、1つ以上の指標を用いて最良のモデルを見つけようとします。まず評価指標は、解くべき課題が分類の課題なのか回帰なのかの2種類に大別されます。次に、より細かく評価指標を選択していくために、いくつか考慮すべき事項があります。以下は有名で非常に基本的ないくつかの評価指標です。実際には、ユースケースに従い、さまざまなものが存在します。以下は評価指標のほんの一部であり、すべては網羅されていないことに注意してください。

1. **F1スコア**：一般的に分類用に用いられる評価指標です。F1スコアは、複数のラベルを分類するのに適した指標です。特に、F1スコアは、精度（Precision）の判定では意味をなさないような、テストセットの大部分が1つのクラスに偏っている場合に特に有効です。たとえば、1,000個のメールのサンプルがあってメールがSPAMメールか否かを判定したい場合を考えます。メールのうち990個は正常のメールで、10個がSPAMメールだとします。単なる精度を調べる場合、分析アルゴリズムを使わずにすべてのメールに対してSPAM

ではないメールであるとラベリングしてしまえば、99%の確率で正しい判断が下せてしまいます。つまり、データが偏っているがために高い精度のスコアを示してしまうような分類問題クラスを推定する場合、偏りがあるデータのうち少数のデータの特性についても正しい判断を下す必要があるような問題を解くのにF1スコアが役立ちます。具体的には、精度（Precision）だけを見るのではなく再現率（Recall）とのトレードオフを評価します。F1スコアでは、精度だけが高く再現率が低いモデルであればスコアが低くなり、精度が高くなくても再現率が高いモデルであれば、相対的にスコアが上がるようにバランスを取ります。F1スコアについてのさらに詳しい説明は、英語のWikipediaページ（下記URL）にあるので参照してみてください。

```
https://en.wikipedia.org/wiki/F1_score
```

2. **AUC**（Area under the Curve）：AUCとは、2つのクラスしかない二値分類用に使用されます。これは、サンプルが正しく分類される確率が、サンプルが誤って分類される確率よりも高くなるか否かを調べるのに役立ちます。AUCについて解説した英語のWikipediaページのURLは以下のとおりです。参照してみてください。

```
https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve
```

3. **RMSE（二乗平均平方根誤差**：Root Mean Squared Error）：RMSEはモデルの出力と正しい値の差を表現するものです。これは回帰モデルが真の出力からどれくらいずれているかを評価するのに用いられます。RMSEについて解説した英語のWikipediaページのURLは以下のとおりです。参照してみてください。

```
https://en.wikipedia.org/wiki/Root-mean-square_deviation
```

2.5 モデルのサービス展開（デプロイ）

モデルのサービス展開はしばしばドキュメント化されていたり、通常業務の一環として説明や検討がなされていたりするものです。チーム内でサービスを構築する際に、モデルがどのように利用されるかについてよく話し合う必要があります。具体的な利用状況を理解することにより、構築したモデルが汎用的に利用可能な状態にならずに終わるといった失敗を避けることができます。汎用的なモデルをサービスに展開することは、ビジネスの価値を生むうえで非常に重要です。

モデルのサービス展開は図2-15の破線内に相当し、AI処理フローの最終フェーズに位置します。

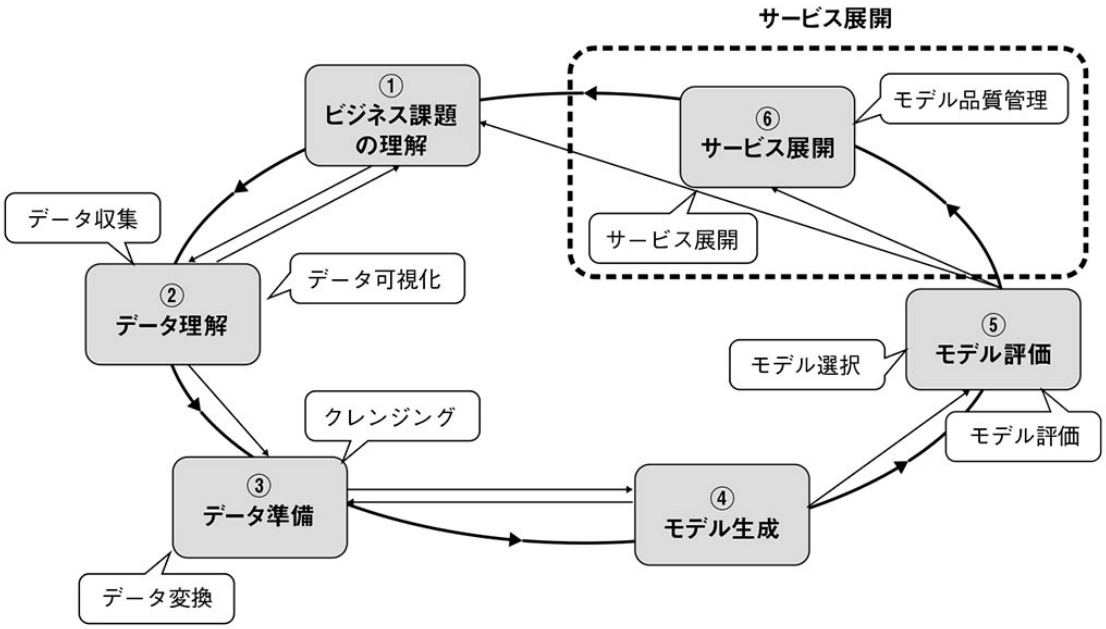


図2-15：サービス展開

本節では、モデルを使ってサービスに展開するための要件やその方法について説明します。

▶2.5.1 サービス展開の要件

最初に、サービスに展開するための要件を正しくチェックするべきです。最も重要なことは**サービス品質保証（SLA：Service Level Agreement）**を確認することです。たとえば、「期待される応答時間」「バッチでサービスを提供するのかリアルタイムなのか」といったサービスの利用条件などがSLAの項目になります。なお、SLAについてはのちほど説明します。

▶2.5.2 モデルの公開方法

モデルを商用サービスとして利用できるようにするには、モデルを公開する必要があります。ここではモデルを公開するために検討すべき項目について説明します。こちらもすべての内容を網羅していないため、その点はあらかじめ注意してください。

1. モデルのREST APIを用いたマイクロサービス化：商用環境でモデルを利用可能にするために、フロントエンドのサービスであるモバイルアプリ、その他のWebアプリケーション、ジョブスケジューラなどのさまざまなクライアントと連携させます。その際、個々の小さな単位のサービスを連携していく**マイクロサービス**という手法で構築していくようにしてください。具体的には、REST APIなどを用いて他のアプリケーションと広範に連携しやすくすることを推奨します。
2. モデルのアプリケーションへの組み込み方法：モデルをフロントエンド側のサービスであるモバイルや、デプロイされる側のWebアプリケーションのようなアプリケーションへ正しく組み込めるように、フロントエンドのサービスで利用されているプログラミング言語や組み込み方法を把握してください。
3. サービスを実行するハードウェアの実装：ハードウェアの性能はサービスの応答性能などに直結するため、アプリケーションの計算をCPUで実行するか、CPU以外の特定用途のハードウェアへモデルを実装するかを検討して

おくことが重要です。

▶2.5.3 フロントエンドサービスの要件

以下にフロントエンドサービスを提供するために必要な検討項目をリストしました。参考にしてみてください。

1. 堅牢性：モデルサーバがダウンした場合に何が起こりますか。冗長性はありますか。
2. SLA：負荷がかかった状態で新たなトランザクション処理の要求が発行された場合、モデルサーバが応答するのにどれくらいの時間がかかりますか。それはどうあるべきですか。またそれはなぜですか。
3. オンプレミスかクラウドのどちらですか：多くの場合、パブリックネットワークを利用する必要はないかもしれません。クラウドでの利用が必要ない場合には、頭が痛いセキュリティ問題の多くから解放されます。セキュリティゲートウェイの内側にモデルを置くことを考えておくと、外部からの侵入を適切に防ぐことができます。
4. 入力データのプライバシー：モデルに対してデータをどのように送信しますか。最低限SSLで暗号化された状態のセキュアな環境にしておくことが推奨されます。
5. モデルのメモリ使用方法：一般的にモデルのメモリ使用方法や使用状況を確認することは重要です。ニューラルネットワークのフィードフォワードパスのような計算を実行する際には、多くのメモリを消費します。サービス展開する前にモデルのプロファイル用のメモリ容量を確保してください。
6. バッチ対リアルタイム：モデルは、新たなデータに対して事前に計算された結果だけを使用する場合があります。そのときには事前の計算結果が蓄積されているデータベースから結果を取り出します。リコメンドを行うシステムの多くは、このような仕組みで動作します。その場合、モデルの出力結果を蓄積す

るオフラインのバッチモードを利用することが推奨されます。

7. 高負荷を想定したうえでの実装方法：逐次、利用可能なリソースのキャパシティをモニタリングして、動的にリソースのスケールを変えるような仕組みを導入してください。内部動作によってタスクの負荷が大きくなりすぎたら、その仕組みを使ってリソースを増やすようにしてください。

▶2.5.4 クラウドへのサービス展開

ベンダーが運営しているクラウドにサービスを展開する場合には、一般的には以下のようないくつかのサービス形態のうちいずれかを利用することが可能です。

1. PaaS（Platform as a Service）形式
2. マネージド・コンテナサービス
3. Kubernetesでのホスティング
4. クラウドベンダーのサービスプラットフォーム

▶2.5.5 オンプレミスへのサービス展開

オンプレミス（オンプレ）にサービス展開するというのは、一般的に自分たちでパッケージを管理してITチームと連携してモデルを構築し実行できるようにすることを意味します。ITチームが要求に応えてくれているうちはこのやり方はうまくいきます。悪くないアプローチです。

オンプレミスにサービスを展開する場合、パッケージ化するために自分たちで自由に多様なツールを選択して使うことができます。ただし、ある程度利用条件に制限がかかってしまい自由度が若干劣るホスティングの環境でサービスを展開する場合は、環境のオーナーと連携して、セキュリティ、運用時間、アプリケーションで必要となるリソースの確保などについて理解を得ながら作業を進める必要があります。

▶2.5.6 クラスタへのサービス展開

クラスタは、クラウドやオンプレのいずれの環境でも実装できる仕組みではありますが、クラスタ環境にすることで、モデルのフェイルオーバーや動的なリソースをスケールさせることが可能になります。クラスタ環境でのサービス展開は、サービスの利用者が非常に多い場合に重要な要件になります。

クラスタを実行する際には、以下のようなさまざまな形式のミドルウェアが含まれるため、それらの仕組みについての知識も必要になります。そのため、このようなインフラに詳しい人と連携してこれらの仕組みを構築していくことが重要になります。

1. （Zookeeper/etcdのような）クラスタ管理
2. （Kafka/RabbitMQのような）メッセージキューによるデータの送受信（pub/sub）
3. （同じモデルで冗長性を確保するための）ロードバランシング
4. 外部からのアクセス（進入や遅延の制御）用のゲートウェイ
5. ロールベース（役割ごとにアクセス権などを付与）のアクセスコントロール
6. （モデルの複数のコピーを展開したり高負荷でアプリケーションを実行したりする場合の）動的なスケーリング

▶2.5.6.1 Kubernetes へのサービス展開

近年、企業では新たなサービスやアプリケーションをすばやく市場に投入するために、コンテナ型の仮想環境である"Docker"と、コンテナの自動運用管理（**オーケストレーション**と呼ばれます）を行うための"Kubernetes"を組み合わせて利用するケースが増えています。読者の方でも今後このような環境でサービス展開に触れる機会が増えると思いますので、これらについて簡単に触れておきます。

仮想化とは1台の物理的なサーバ上で複数の仮想的なマシンを構築／運用する技術です。仮想化の手法にはさまざまなものがありますが、近年、コンテナ型仮

想化として**Docker**という技術が注目され非常に多く使われるようになってきています。

コンテナという名前がついていることからイメージできると思いますが、コンテナでは、アプリケーションを動かすために必要な機能をパッケージングします。つまり、Dockerでは、ソフトウェアを動かすために必要なライブラリ、システムツール、コード、ランタイムなどをパッケージングして展開することができます。

単一の物理サーバ上でコンテナを展開するのはDocker単体でも容易に実現可能ですが、複数台のサーバで制御しなければならないクラスタ環境でコンテナを運用するには、コンテナの起動／停止などの操作だけでなく、物理サーバ間のネットワーク接続やストレージの管理、コンテナを動かす物理サーバのスケジューリングなどさまざまな機能が必要になり、独自で行うのは非常に厄介です。

通常、Helmというソフトウェアを使って、クラスタ環境のための作業をDocker自体が持っている機能で設定していくことも可能です。ただし、それだけでは依然として運用が複雑であるため、コンテナの自動運用管理として近年Kubernetesが広く利用されるようになっています。

Kubernetesには、クラスタ上でのサービスの展開や、コンテナの管理を容易にするためのツールとして、最近ではHelmが実装されています。Kubernetesでのクラスタ構築を検討している場合には、運用管理を容易にするためにHelmを利用することをお勧めします。

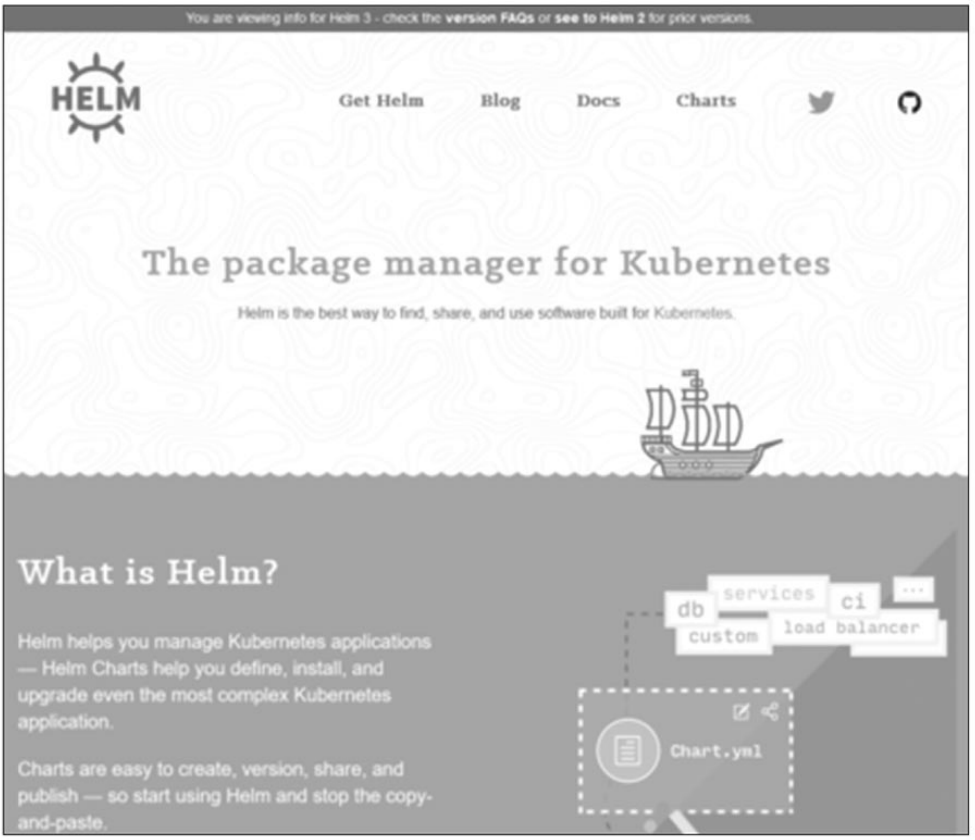


図2-16:Kubernetesのパッケージマネージャ（Helm）

Kubernetesを用いてクラスタ環境を運用管理するための注意点についても簡単に紹介します。Kubernetesでは、コンテナをPod（ポッド）という単位で管理しています。これはDockerの管理単位と読み替えていただいてかまいません。特に、アプリケーションを商用サービスで動かすために何のソフトウェアが必要になるのかを事前に検討する際には、実装が想定されるすべてのソフトウェアを、Dockerの仮想環境内である1つのポッド内に展開すべきです。同じポッド内に展開しておくことで、適切にクラスタを分離できて、自己完結型のネットワークを持たせることが可能になります。

Kubernetesにサービスを展開する場合、通常、さまざまな要件を満足する処理を実現するためにKubernetesのエコシステムを使用します。それぞれの要件によって利用するエコシステムは異なるため、詳しい使い方を知りたい方は、別途Kubernetesの書籍などを参照してください。

▶2.5.6.2 クラスタのアーキテクチャ概要

ここで参考までに、ディープラーニングのモデルの分散学習を実行するためのクラスタについて紹介します。なお、この解説は、マイクロソフト社のWebサイト※1の情報に基づいています。具体的には、このクラスタのアーキテクチャにおける分散学習の手順とアーキテクチャを構成するコンポーネントを説明していきます。

◉――分散学習手順

分散学習の手順は、以下のようになります。

- クラスタで実行するための訓練用スクリプトを作成し、モデルを学習したのち、ファイルストレージに転送します
- Premium Blob Storageにデータを書き込みます
- Azure Machine Learningワークスペースを作成します。ここで、Dockerイメージをホストで利用するためのAzure Container Registryも作成されます
- Azure Machine Learning GPU Clusterを作成します
- 実行ジョブを送信します。固有の依存関係を有するジョブごとに、新しいDockerイメージが作成され、コンテナのレジストリにプッシュされます。実行時には、適切なDockerイメージが動作し、スクリプトが実行されます
- すべての結果とログはBlobストレージに書き込まれます

以上が分散学習の実行手順です。次にアーキテクチャを構成するコンポーネントについても簡単に紹介します。

◉――アーキテクチャを構成するコンポーネント

このアーキテクチャは、次のコンポーネントで構成されています。

※1 『Distributed training of deep learning models on Azure』(https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/ai/training-deep-learning)

- Azure Machine Learning Compute

Azure Machine Learning Computeは、ディープラーニングのワークロードに対応した仮想マシンをサポートしています。必要に応じてリソースを増やしたり減らしたりする機能を有し、アーキテクチャの中心的な役割を果たします。Azure Machine Learning Computeは、仮想マシンのクラスタのプロビジョニングと管理、ジョブのスケジューリング、結果の収集、リソースのスケーリング、障害の処理を支援するサービスです。

- Blob Storage

ログと結果の保存に使用されます。

- Premium Blob Storage

訓練データの保存に使用されます。Premium Blob Storageは、Blobヒューズと呼ばれるBlobストレージ用の仮想ファイルシステムドライバーを使用してノードにマウントされます。Premium Blob Storageは標準Blobよりも高い性能を提供し、分散学習での使用が推奨されています。データはローカルに保存するためにマウントされたBlobヒューズで、キャッシュを使用します。この機能により、ローカルストレージからデータを読み込めるため、後続のエポックで最も高い性能を実現するためのオプションになります。

- Container Registry

Azure ML GPU Clusterが訓練を実行する際に使用するコンテナ用のレポジトリで、Dockerイメージを格納するために使用されます。

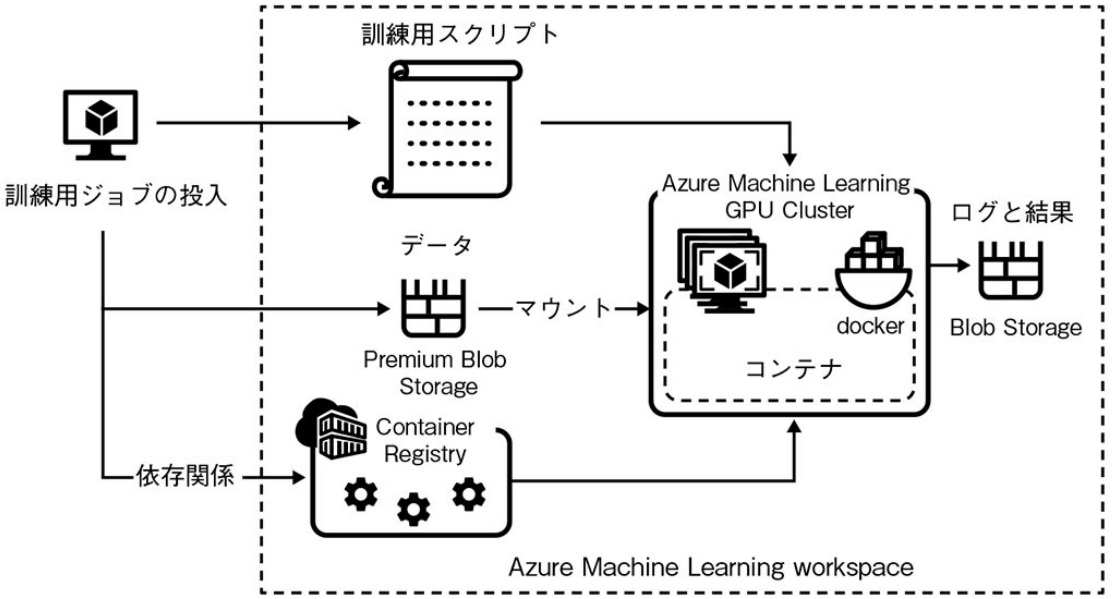


図2-17：分散学習を実行するためのクラスタアーキテクチャの例（出典先の図をもとに制作）—［出 典］『Distributed training of deep learning models on Azure』（https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/ai/training-deep-learning）

▶2.5.7 ハードウェアの選択

機械学習を実行する際には、通常のCPUでは性能が不足するため、より多くの計算量をこなせるハードウェアが必要になる場合があります。必要なハードウェアのスペックは、分析のシナリオがどういったものなのか、入力データのサイズはどのくらいかなど、具体的な条件に依存しますが、単一のデスクトップのパソコンではなく、より大規模なサーバが必要になることがあります。PoC（概念実験）を実施する場合、一般的に全データセットを使用しないため、大きなハードウェアは必要ありません。PoC後に使う実際の商用環境を用意する場合には、その要求事項を確認し、計算時のコストがどれくらいになるかがわかったら、どのようなハードウェアが必要なのかを再検討します。ハードウェアと一言で言っても単なるラップトップから大規模サーバまでさまざまです。通常、コストを最適化するには、ハードウェアにかかるコストを最小限に抑えることで他の領域でかかるコストに影響

を与えないようにしていきます。

ハードウェアのキャパシティプランを検討する場合は、予算がどのくらいあるのか、実際にSLAを遂行するのにどのようなサーバが必要なのかを確認しておくようにしてください。この計算は投資対効果に従って算出されるべきで、決してコストを抑えるためであってはなりません。高性能な環境を実現することでより良いユーザ体験が得られ、サービス品質の向上につながります。チームとしては、そのような投資に対応する準備が必要になります。

▶2.5.7.1 特定用途のチップの利用

GoogleのTPU（Tensor Processing Unit）やNVIDIAのGPUなどのさまざまなチップについて耳にしたことがあるでしょう。これらのチップは独自のメモリ（RAM）や特定の計算方式を搭載しており、それぞれトレードオフがあります。チップがどれだけのRAMを搭載していて、実行したいタスクにおいてCPU使用率を最大限に活かせるかについて理解しておきましょう。

読者の方々がサービスを展開するプロジェクトにおいて、特定のチップが必要か否かを評価するときに、自身のプロジェクトの特定の課題に対して、より良い性能が出せるのかどうかを理解しておくべきです。チップで必要となる最低限のスペックを定義しておくのが役に立ちます。利用を想定しているチップが要求事項を満たせるかを正確に知るには、想定しているチップ以外にもいくつかの特定用途のチップをベンチマークしておくことをお勧めします。特に大きな画像で画像のワークロードを実行する場合、一般的にはGPUを使用するとより良い性能が得られます。

小規模のカラム型のデータの場合には、1万個以上のサンプルを一度にスコアリングするといったことをしない限りは、一般的にはGPUは必要ありません。

ただし、最初にプロジェクトを遂行していく際には、特定用途のチップについて気にしすぎないようにすることをお勧めします。深刻な性能のボトルネックに遭遇した場合には、ボトルネックが何か、そのボトルネックがCPUに関連しているか否

かを理解し、必要に応じて特定用途のチップの採用を検討してください。もし性能が足りない場合には、まずGPUの採用を検討することをお勧めします。

初めて機械学習を進める場合は、できるだけCPUだけを使用してプログラムを実行してみてください。ただし、最近は多くのディープラーニングのフレームワークでGPUが使いやすい形でサポートされているので、モデルを訓練する際にGPUで動作させるのは簡単です。

▶2.5.7.2 デフォルト値の利用

近年の機械学習のライブラリにおいてデフォルト値は、ある程度の経験に基づき最初に試すべき値に設定されている場合が多くなっています。機械学習ライブラリの特定の機能を最初に実行する場合、特にそのライブラリに知見がないときには、デフォルト値で試してみることをお勧めします。

▶2.5.8 サービス展開の準備

モデルのサービス展開の準備が整ったら、サービスに含まれるすべてのさまざまなコンポーネントを調べる必要があります。モデルのサービス展開で必要となるコンポーネントは、一般的には以下のものが含まれます。

1. データパイプラインもしくは前処理
2. モデルファイル：pickle化したデータ形式か、Protobufやhdf5のようなフォーマットになっています。モデルによってはこのファイルは巨大なサイズになります。
3. 出力処理：機械学習で得られた出力形式ではユーザが理解しづらい場合があります。その場合、必要に応じて、元のデータよりも解釈しやすい形にデータを変換するために、新たなデータパイプラインまたは別のモデルを利用して、期待される出力形式にデータを変換します。これらの形式は、オリジナルで利用されるモデルの出力よりも、さらに解釈しやすい形にするようなシンプ

ルな変換マップのようなものです。たとえば、モデル出力の確率を文字列のラベルに変換するなどがこのケースにあたります。

通常、これらのコンポーネントの資産に対するバージョン管理や、最新バージョンを実装後に不具合が発覚した場合を考慮し、ダウングレードの戦略を取れるようにしたくなるはずです。そこで、実行した経験や実行結果を記憶しておくためにノートブックをストアするように、Zip形式のファイルやバイナリ形式にして、モデルの作成時に生成される中間結果などをバージョン化して保存します。

▶2.5.8.1 GDPRの取り扱い

GDPR（General Data Protection Regulation：**一般データ保護規則**）とは、ヨーロッパ連合が導入する、個人情報、個人データの扱いに関する規則ですが、機械学習においてGDPRの取り扱いは、かなり一般的な問題になってきています。問題になる理由は、GDPRには個人のプライバシーの取り扱いが含まれており、機械学習はユーザデータを使って、ビジネスでの利益につながる振る舞いやパターンを見つけようとするため、個人のプライバシーを適切に取り扱う必要が生じています。

GDPRの準拠については本書の対象範囲外ですが、プライバシーと機械学習の問題を考えるうえでいくつかのヒントについて以下に手短に説明します。

1. どのユーザデータが使用されているかを常に知っておくこと。
2. モデルを汎用的に利用できる堅牢なものにするのと同時に、セキュリティのためにデータを匿名化してください。
3. データサイエンティストのチームが必要以上のデータを閲覧せず、チームがGDPRの問題を正しくうまく処理するために、極めて具体的なプライバシーのトレーニングを行うことを保証してください。
4. 整合性を備えた正しいプライバシー文化を持つことが重要です。成功のカギ

としては、経営陣が陣頭指揮に立ち、プライバシーの保護にコミットする必要があります。

5. プライバシー情報が含まれるユーザ経験に関するデータをストアする場合、データを商用環境からできる限り遮断するようにしてください。どのデータがいつ使われたかについて把握できるように、バックアップもモニタリングしてください。古いアーカイブからデータが漏洩するのは非常によくあることです。バックアップのデータについても暗号化してください。

▶2.5.8.2 ソフトウェアのバージョン管理

ソフトウェアのバージョン管理を行うには、いくつかの形式の異なるストレージが必要です。ソフトウェア資産としては大まかに以下のものがあります。

1. メタデータ。通常はリレーショナルデータベース内に存在します。
2. ソフトウェア、データおよび設定情報。それらのソフトウェア自体の資産を保存するためのBlob形式のバイナリのストア（tarまたはzipアーカイブなど）が必要です。

新たなモデルをサービス展開するとき、いつ展開されたか明らかにするためにタイムスタンプを付けてアーカイブにしてバックアップを取ってください。テストの精度情報などのメタデータは、モデルを展開する場合に便利です。これについてはメンテナンスの節で詳しく説明します。

▶2.5.8.3 モデルのサービス展開手順

モデルをサービスとして展開する場合、モデルを組み込むためのコンポーネントをアプリケーションにパッケージングするのが一般的です。モデル管理専用のインフラストラクチャを使って構築するか、クラウドのような環境で整備するかの選択は、実際にサービスのオペレーションを行うチームに処理を委ねるべきです。

モデルをアプリケーションの一部としてパッケージングしサービス展開するためのステップは以下のようになります。

1. 新たなアプリケーションの展開
2. モデルサーバの停止
3. Blob形式のアーカイブを使って新しいモデルをストレージにストア
4. 一時的にサービスのリクエストをブロックするか、もしくはインフラストラクチャによってはフェイルオーバーさせる
5. 展開される新たなバージョンのモデルをサービスに反映するために、メタデータをアップデート
6. 新たなバージョンのモデルを使ってサービスをスタート
7. 必要に応じて、新たなモデルが正しく反映されているかどうかを確認するために、モニタリング機能を用いてアップデートされているかを確認

▶2.5.8.4 データ処理方式

機械学習では、多量のデータ処理が必要となります。ビッグデータという言葉が流行しだした当初は、実行時間が非常に長くかかる大量のデータ分析を定期的に行うバッチ処理を行っていました。近年は、ストリーミングデータを使った短い応答時間が求められるリアルタイムのデータ分析でも機械学習が利用されるようになってきました。その結果、リアルタイムでの分析処理が必要な場合、バッチ処理とリアルタイム処理を組み合わせたシステムアーキテクチャを構築する必要があります。

バッチ処理とリアルタイム処理を組み合わせたシステムアーキテクチャの代表例が、**ラムダアーキテクチャ**と呼ばれるものです。ラムダアーキテクチャは、リアルタイムのアップデート処理と機械学習処理を機械学習のデータパイプライン処理として実行します。ラムダアーキテクチャとは、そもそも大容量のデータに対してバッチ処理とリアルタイム処理の両方を効率的に実行できる分析用データ基盤を構

築するために用いられるアーキテクチャのことです。

このような仕組みを実現するために、ラムダアーキテクチャでは以下のような条件下で実現することを仮定しています。

すべての処理はアペンドのみ（追加処理のみであり、変更や削除などが含まれない）で実現します。これはすべてのレコードでアップデートが発生することがなく、新たなレコードが生成され続けるだけです。バージョンはいくつかのデータとタイムスタンプで構成され、タイムスタンプは常に前のレコードのタイムスタンプよりも新しいものになります。これはログと呼ばれます。

上記のような条件をもとにラムダアーキテクチャを用いて機械学習のパイプラインを構築することにより、リアルタイムで機械学習のモデルを利用可能とする堅牢なエンドツーエンドのアーキテクチャが実現できます（リアルタイムでの機械学習はオンライン学習と呼ばれることもあります）。

2.6 モデルの品質管理

出来上がったサービスを維持していくには、品質管理が必須になります。そのため、AI処理フローのサービス展開には、モデルの品質管理を含めることが重要です。モデルの品質管理は基本的にモニタリングとメンテナンスで構成されます。

▶2.6.1 モデルのモニタリング

最初にモデルの品質のモニタリングについて説明します。モデルのモニタリングでは、遅延などのインフラストラクチャ側の指標に加え、モデルがどれだけ実社会でうまく機能しているのかについてもチェックする必要があります。そのため、モデルが統合されているシステム全体に関する情報もモニタリングする必要があります。

以下にモニタリングすべきそれぞれの指標を簡単に説明し、のちほどその概念をさらに詳しく説明していきます。

▶2.6.1.1　インフラストラクチャの指標

インフラストラクチャの指標には、遅延、リクエストの失敗、メモリの利用状況などがあります。

与えられたモデルには、それぞれに固有の指標があります。モデルが期待された負荷で稼働しているかについて、常にベンチマークを行って確かめてください。ユーザを長時間待たせ続けてしまうようなモデルは一般的に有効ではありません。負荷とユーザの体感とのトレードオフは常に起こります。たとえば、90年代の話になりますが、Netflixが、リコメンデーションエンジンの優れたモデルに100万ドルの賞金を支払うというコンペティションが行われました。そのときの優勝者には100万ドルの賞金が支払われましたが、その後の実用化は非常に限定的な領域にとどまる結果となりました。というのも、優勝者のモデルはリコメンデーションの精度としては優れていましたが、すべての計算を完了するのに時間がかかりすぎて、さらには非常にメモリ容量を要するものだったからです。

このような問題が発生しないようにするには、インフラストラクチャの指標についてもしっかり検討するようにしてください。

▶2.6.1.2　機械学習のモデルの品質検査指標

機械学習のモデルをサービス展開する場合にも、通常の商用アプリケーションと同様に何らかの品質検査が必要となります。機械学習のモデルの品質検査は、テストのデータセットを使って長期間にわたりモデルの精度をモニタリングすることで実現できます。

モデルをサービス展開するときに商用専用のテストデータセットを準備し、毎回そのデータをもとに評価を実施するようにしてください。テスト専用のデータセットを準備することで、新たなモデルでサービス展開を構築するときに、以前のサービスと比較して精度が落ちていないことを確認できるため、安定した品質が得られます。

この品質検査は継続的統合（Continuous Integration）からヒントを得てい

ます。継続的統合とは、新たなアプリケーションを商用環境に展開する前に、いわゆるアクセプタンステストと呼ばれる品質検査の実施を行う仕組みを実装することで、アプリケーションを継続的に商用環境に統合していくことが可能になります。機械学習のモデルの実装に継続的統合の概念を当てはめた場合、そのアクセプタンステストとして、上記のような機械学習のモデルの精度を評価する仕組みをAIの処理フロー内に組み込んでください。そうすることで、品質検査が確実に、かつ、非常に簡単に実現できるようになります。

モデルの品質をモニタリングするには、常にこれらの指標を収集し可視化などを行って把握する仕組みが必要になります。これを実現するにはさまざまな手法がありますが、たとえば、オープンソースでかなり利用が進んでいるPrometheusやGrafana（図2-18）のようなツールを使用することで簡単に実現できます。一般的に、特定の時間のモデルの動作を理解するには、時系列で指標をモニタリングできるようにする必要があるので、時系列のデータベースなどを利用可能なツールの使用を推奨します。

図2-18:Grafanaの可視化例―［出典］Grafana公式ページのライブデモからの引用（https://grafana.com/grafana/）

▶2.6.2 SLAのモニタリング

SLAを正しくモニタリングするには、最初にSLAの定義を行い、その後、SLAを満たしているかの確認を行います。そのために必要なモニタリングの手法について説明します。

▶2.6.2.1 SLAの定義

SLAは解決すべき課題によって大きく異なりますが、一般的に以下の指標項目は押さえておくべきです。

1. サービスの遅延の許容時間
2. サービスの稼働時間
3. サービスの精度
4. サービスの修正が送信された回数

▶2.6.2.2 SLAの典型的な仕組み

商用の環境で一連のサービスレベル指標が維持されているかを確認するために、さまざまな指標が用いられます。それぞれの指標の目標値が満たされているかを確認することで、サービスレベルを保証することができます。

それぞれの指標がSLAの条件を満たしているかを確認できるようにするために、既存のインフラストラクチャに対してそれぞれの指標の値をモニタリングし、条件が満たされなくなった場合にアラートを出力するようにします。

クラウドサービスを利用する場合、クラウドベンダーのSLAは常にサービスの稼働時間をカバーするものです。クラウドでのサービスのSLAは通常、エンドユーザが導入するサービスによって異なるため、クラウドサービスの導入時に、達成すべきSLAのそれぞれの指標の条件がクラウドサービスのSLAで満たせるのかどうかチェックしてください。たとえば、クラウドベンダーのSLAでは、常にクラウドベンダーが提供するサービスに稼働時間の項目が示されており、クラウドベンダーが

保証する稼働時間が、あなたが実装するサービスの稼働時間に影響を与える可能性があります。そのため、クラウド利用時にはクラウドベンダーのSLAの範囲でシステム停止が起こることを前提に準備をしてください。

▶2.6.2.3 SLAを満たしていない場合の対処方法

求められているSLAが満たされていないと発覚した場合、最初にすべきことは、それぞれの指標を見ることです。その前提としてSLAを満たさない個所が問題ごとにきちんと細分化できている必要があります。問題個所が細分化されている場合には、それぞれの状態をチェックして、問題がある指標を特定することで、問題の根本原因を突き止めていくようにしてください。

たとえば、遅延が起きるという問題が生じていた場合、それはインフラストラクチャの問題に帰結します。精度に問題が発生していた場合には、データサイエンスの問題です。

上記のように、それぞれの問題と指標の関連性をできるだけ明確に定義し、事象を正確にとらえるためにモニタリングすることで、多くの予期しない障害に対しても早期に事象をとらえて被害を軽減できるようになります。

▶2.6.3 モデルのメンテナンス

モデルのメンテナンスとしては、モデルをITの資産の一部だと考えて、適切にバージョン管理を行う必要があります。アプリケーションと同じような方法で、モデルをテストできるようにしていきましょう。

モニタリングと改善を継続することにより、モデルの利用可能な状態を持続できます。モデルを適切に運用していくには、品質などに関する適切なポートフォリオを持つ必要があります。この仕組みにより、構築すべきモデルがどれなのかをチームが理解しやすくなります。それぞれ実装されたモデルごとに、何がどこに構築されているかなどのポートフォリオに関する情報を使って、モデルの品質を維持するためにクリアすべき明確な基準を作ります。そのうえで、それぞれのモデル

の精度などの評価結果をモニタリングおよび評価することで、課題を再検討するか、ビジネス上の優先度に合った新たなモデルに移行するかなどを考えていきます。

▶2.6.3.1 メンテナンスで生じる問題

通常、すべてのモデルを蓄積し、モデルをどれくらいの期間アーカイブにしておくのかを把握し続けることは、非常に骨の折れる作業です。モデルの追跡をアプリケーション開発のライフサイクルに同期させ続けることも、大変な作業です。インフラストラクチャの統合を確実なものにするために、アプリケーションと、開発されたモデルの両方を常に自動的に同期させるようにしてください。

▶2.6.4 コンセプトの不安定さのモニタリング

課題のコンセプトの不安定さとは、課題で用いられている入力データと出力データの関係が、時間や外部影響によって変化することを意味しています。課題の数学的なコンセプトの不安定さは、複数の事象が複合的に重なって起こっています。一般的に、指定された入力のサンプルデータの移動平均値と精度を組み合わせて長期間にわたってモニタリングすることにより、それらの数値の変化をもとにコンセプトの不安定さが生じているか否かをチェックすることがある程度可能になります。また、入力データに対して、特定の形式で正規化などを行うことで、これらの変化に対してモデルの精度が保たれるようなデータパイプラインを構築することが可能になります。平均や標準偏差の値は時間の経過に伴い変化するかもしれないので、scikit-learnのStandardScalerのように平均や標準偏差を使ってデータの縮尺を変えるような標準化の機能が実現できるのであれば、それらの機能を利用しましょう。入力データなども時間に伴って変化が生じるため、モデルだけでなくデータの標準化機能（ノーマライザー）についても更新するようにしてください。

▶2.6.4.1 コンセプトの不安定さが生じる代表例

特定の領域、特に時系列のユースケースに代表されるように一部の領域では、課題のコンセプトが常に変化していきます。具体的な例としては以下のようなものがあります。

1. 不正検知
2. ネットワークセキュリティ
3. 送電網のモニタリング
4. ユーザ行動を理解するためのWebサイトのトラフィックのモニタリング
5. スパムのモニタリング

ここで重要なのは、母集団が時間経過に伴って変化していないかを確認することです。コンセプトの不安定さは、どのような属性からでも生じる可能性があります。望ましいのは、可能なら属性ごとに変化していないかをモニタリングすることです。

▶2.6.4.2 コンセプトの不安定さのテスト方法

精度と属性の両方に対して過去の値と現在の値について分散の違いなどを比較してモニタリングすることで、領域のシフトがいつ発生しているのかを確認します。通常、ある程度の分散は許容されるべきですが、危険水域に達する分散量の差がどのくらいなのかを、常にアプリケーションごとに決定するように努めてください。

▶2.6.4.3 コンセプトの不安定さと戦う方法

コンセプトの不安定さと戦うには、本章全体で語られている手法を使用しながら広範囲にわたりさまざまな指標をモニタリングする必要があります。コンセプトの不安定さのモニタリングにおいて、一般的に、商用で使うタスクがうまく動作し

ない原因を特定するには、その原因が特定の精度や属性のドメインで分散の変化によって生じたものか否かを調べます。

あなたの組織でコンセプトの不安定さがどの程度重要なのかは、解決すべき課題の特性によりますが、堅牢なサービス展開が可能なインフラストラクチャと評価指標を定常的にモニタリングできる仕組みを実装しておくことで、多くの問題を解決することができます。

▶2.6.5 オンラインでのモデルの再訓練

オンライン（リアルタイム）でのモデルの再訓練は次のように定義されています。現在、稼働中のモデルを少しずつ更新してダウンタイムをゼロにしつつ、新たなデータでもクリアすべき性能などの基準を達成することを確認しながら、継続的な改善を行っていくことです。

このような仕組みを実現するのは極めて難しいことです。こうした仕組みを実装する場合、通常、次のようなインフラストラクチャの問題が発生します。

1. 稼働中のダウンタイム回避とモデル更新に対して適切に対応できない
2. 更新されるべきモデルのローテーションがうまく回らない
3. 新たなデータセットを用いた訓練が望ましいものかを知る方法が確立されていない
4. モニタリングの後、モデルと一緒に展開された特定のテストで正しく十分な精度がでないと判断された場合に、スムーズにモデルをロールバックする方法が存在しない
5. インクリメンタルな（増分の）更新を実現するために、モデルに送信されるデータをバッチ処理するための方法が確立されていない。たとえば、画像を用いた訓練の最中で、特定の数の画像がキューに入ったら計算が行われるようにするには、画像のバッチ処理を行う必要がある
6. モデルに送信されるデータセットが誤っているか、バランスが悪いものかを知

る方法が確立されていない。バランスの悪いデータセットはモデルの精度に影響を与えてしまう

7. オンラインの再訓練を実装する方法が確立されていない

データをオンラインで収集し分析に利用するには、標準のREST APIやKafkaやRabbitMQなどのメッセージキューを用いることで、入力データをリアルタイムに収集する仕組みを実装する必要があります。メッセージキューとは、システム間でデータの受け渡しの仲介役を果たし、データを一時的に保存（キューイング）する仕組みのことです。特にメッセージキューを持つ仕組みでは、メッセージキューにおけるデータ収集の仕組みであるアキュムレータが、メッセージキューに入力データを蓄積していき、一定量になるか、一定時間が経過したら、それらのデータをモデルに送って再学習させるような仕組みが実現できます。メッセージキューの仕組みを用いることで入力のデータが十分に蓄積できた場合、モデル更新と再配布を目的とした別のメッセージキューの機能を実装することにより、再学習とモデル更新／再配布とを別々に処理することも可能になります。さらに、メッセージキューでは、データが機械学習の入力データ形式になっていない場合、キューにデータを蓄積している最中に1つ以上のデータ変換を行わせるようなことも可能です。

▶2.6.6 A/B テスト

本章で示したアイデアに加えて、A/Bテストのようなテクニックを使って商用環境で実験を行うこともできます。それには、異なるチーム同士が連携してうまく機能することが必要です。

商用環境でモデルにA/Bテストを行う目的は何でしょうか。2つの異なるモデルを用意して、それぞれに対してトラフィックを切り替えることで、特定のタスクでどちらのモデルの性能が優れているのか／向上するのかをモニタリングしようというわけです。この方法は、リコメンデーションエンジンを構築する際に、一般的に利用されています。リコメンデーションエンジンのKPIは、エンドユーザ向けにアイ

テムを表示することで、ある種のコミュニケーションを促進することです。

A/Bテストをリコメンデーションエンジンに利用することで、さまざまな顧客や利用状況／形式の変化など多様な実社会の変化をとらえ、必要な指標をモニタリングできるようになります。

2.7 プロジェクトの検討項目としてのAIテンプレート

AIのプロジェクトを成功に導くには、これまで説明してきた多岐にわたる項目を順番に検討していくことをお勧めします。そこで最後に、それらの検討項目をAIのテンプレートという形で以下にまとめてみました。実際のプロジェクトを複数経験していくにつれて、独自のテンプレートを作成していくことになるかと思いますが、初めてAIのプロジェクトに関わる方や、AIの活用に試行錯誤されている方の指針として、チェックリスト的に活用いただければ幸いです。

■ COLUMN 【AIテンプレート】

1. ビジネス課題の理解
    - ・チームのスキル要件の定義
    - ・適切なチームの構築方法
    - ・データサイエンティストを雇うタイミング
    - ・ビジネス要件の定義
    - ・ITインフラ要件
    - ・機械学習の採用基準の確立
    - ・課題の設定指針

2. 探索的データ分析（EDA）
    - ・EDAの直感的理解

・データの理解

・EDAのプロセスの全体像

・EDAプロセスで用いられるツール

・評価指標の使用方法

・EDAの繰り返し実行

・パイプラインの不具合の調査方法

・プロジェクト範囲の設定方法

・プロジェクト期間の見積もり方法

・問題発生の未然防止策

・EDA結果の検証手順

3. モデル作成に向けた下準備

・データ準備の最適化

・モデルのアーキテクチャの最適化

・データサイエンスのコンペティションの利用について

4. モデル開発

・経験の実行

・経験の実行ステップ

・評価指標（Metrics:メトリクス）の理解

5. モデルのサービス展開（デプロイ）

・サービス展開の要件

・モデルの公開方法

・フロントエンドサービスの要件

・クラウドへのサービス展開

・オンプレミスへのサービス展開

・クラスタへのサービス展開
  - Kubernetesへのサービス展開
  - クラスタのアーキテクチャ概要
・ハードウェアの選択
  - 特定用途のチップの利用
  - デフォルト値の利用
・サービス展開の準備
  - GDPRの取り扱い
  - ソフトウェアのバージョン管理
  - モデルのサービス展開手順
  - データ処理方式（ラムダアーキテクチャ）

6. モデルの品質管理

・モデルのモニタリング
  - インフラストラクチャの指標
  - 機械学習のモデルの品質検査指標
・SLAのモニタリング
  - SLAの定義
  - SLAの典型的な仕組み
  - SLAを満たしていない場合の対処方法
・モデルのメンテナンス
・コンセプトの不安定さのモニタリング
  - コンセプトの不安定さが生じる代表例
  - コンセプトの不安定さのテスト方法
  - コンセプトの不安定さと戦う方法
・オンラインでのモデルの再訓練
・A/Bテスト

また、上記のコンセプトベースのAIテンプレートから一歩進んだ形で、実際のAI処理フローを簡単に実装するためのオープンソースツールとして、本書の著者の1人であるアダム・ギブソンがkonduit-servingを開発中です。

konduit-servingは、AI処理フローのうち、データ準備から、モデル開発、さらにはモデルの出力を理解しやすい形にデータ変換する後処理までのパイプライン処理を簡単に実装できるようにする仕組みです。

本書のAppendixでは、konduit-servingを用いたサンプルコードの実装例を紹介します。ご興味がある方はぜひ試してみてください。

2.8 まとめ

本章では、チームの構築方法、プロジェクトの範囲の設定方法、運用環境でのサービス展開の全体像に焦点を当て、検討すべき項目について取り上げてきました。これらのベストプラクティスにより、投資対効果が証明されていない経験において時間を浪費することを回避できます。もう1つ学んだことは、AI処理フロー全体の中でそれぞれのステップの全体像を俯瞰しつつ、各ステップをどのように対処すべきかの基本方針についても学びました。これにより、分離された環境の中で単独で作業するのではなく、共同作業を行うために標準的な作業内容を実行できるようになります。つまり、孤立した環境の中で作業を行うのではなく、新たな共創的なチームを容易に構築できるようになるのです。

さらに、本章の内容を理解することで、最初の経験を開始する場合であっても、チームが機械学習を実社会で活用することを意識付けられるようになります。また、企業が成功しながら最小のリスクで機械学習を採用するのに必要なステップの情報を活用しながら、ステークホルダーなどに対して、事業を継続していくために必要な成果報告に値する内容を提供できるようになるでしょう。

第3章

ディープラーニングの基本構成

直感的に仕組みをとらえる

Deep Learning Project Template

ディープラーニングを用いた分析アプリケーションを完成させるには、ワークフロー設計とディープラーニングのモデル設計が必要です。前章でワークフロー（AI処理フロー）設計について説明しましたが、本章では、ディープラーニングのモデル設計に必要となるさまざまな項目について説明します。

内容的には、ディープラーニングのモデル設計で不可欠なディープラーニングの基礎を理解できるようにしていきます。ディープラーニングについては、ここ数年の流行に従い、関連書籍やブログなどを通じて、さまざまな情報に触れる機会が増えてきています。そのため、すでに語られているディープラーニングの歴史や、基本的な構造などの内容について繰り返すことは極力避けます。その代わりに、今まで語られることが少なかった、ディープラーニングの本質的な動作について、直感的に理解いただける説明に注力します。たとえば、ディープラーニングのモデルの中で起こっている作用について、数式に頼らずに、入出力の変化の結果などで見ていきます。

また、ディープラーニングの開発現場で日々経験されていることとして、モデルの構築と評価を進めていくにつれて、ある場合は高い精度のものができたとしても別の場合ではうまく精度が出ない、といったことがあるのではないでしょうか。モデルの作成がうまくいかなかった理由について、すでに考察できる現場であれば問題はありません。しかしながら、そのような知識がないために、無軌道に試行錯誤が繰り返されている開発現場に立ち会うことがあります。

ディープラーニングの動作の仕組みを理解するのは非常に苦労します。ディー

プラーニングではその性質上、細かい内部動作がブラックボックスな状態のままで、自動的に学習が進められていきます。実際、正確にすべての動作を、有限の時間の中で人の頭脳で追うことは不可能です。そのため、内部動作を含めた仕組みの理解を深める情報が比較的に少ないのが現状です。開発現場で無軌道に試行錯誤が繰り返されやすくなっている主な原因の1つがその点にあると筆者は考えています。

したがって本章では、これらの失敗した原因を推察するための手がかりとして、ディープラーニングの仕組みに関する情報についても説明します。これらの情報を通じて読者の方々が、ディープラーニングの開発を行う際に、少しでも考察するための手段が増えるとうれしいです。

本章は、以下の節で構成されます。

それではさっそくスタートしましょう。

3.1 ニューラルネットワークの処理概要

ニューラルネットワークとは、人間の脳の神経細胞（ニューロン）とそのつながりである神経回路網（ネットワーク）を数理モデル化したものです。そのため、はじめにニューラルネットワークの基礎要素である人工ニューロン（以降、ニューロン）について簡単に説明し、その後でニューロンをこの仕組みの発想の原点にあたる脳細胞の機能と比較します。ニューラルネットワークはディープラーニングの動作の基本なので押さえておきましょう。

次の図3-1はニューロンと脳細胞の関係を簡単に示したものです。ニューロンは、重み、総和器、活性化関数、バイアスで構成されています。図の「ニューロン」には「重み」が示されていますが、重みの大小は、情報を伝える割合、つまり情報の伝わりやすさを表しています。総和器は、複数のニューロンからの情報を1つにまとめる役割を果たしています。また、活性化関数は、ノードに集まった入力信号の総和をどのように活性化するかを決定する関数のことです。

ニューロンのそれぞれの機能を脳細胞に例えると、重みの大小は軸索の太さに対応します。また、総和器は細胞体にあたります。なお、活性化関数については、次の節で説明します。

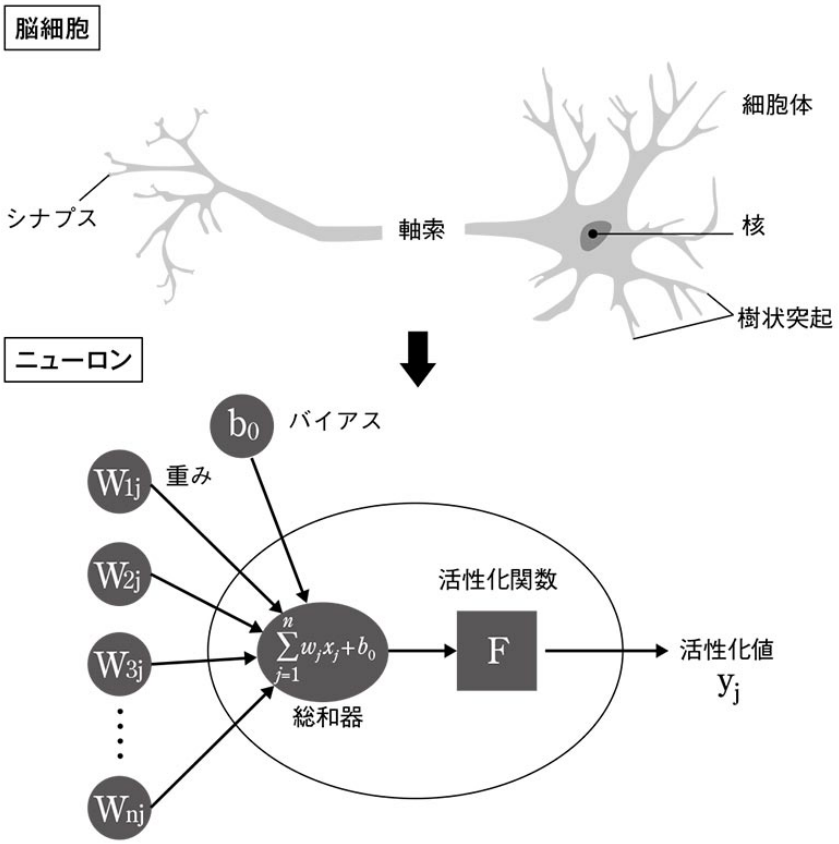


図3-1：神経細胞とニューロンの関係

▶3.1.1 ニューラルネットワークの情報伝達の仕組み

図3-2は、最も単純な1つの入出力データにおけるニューラルネットワークの機能の概要です。この図のとおり、入力データは重みを算入してニューラルネットワークに入力されます。入力データが出力の予測値と関連性が強い場合、重みは大きい値となり、関連性が小さい場合には、重みは小さくなります。たとえば、関連性がまったくなければ重みが0になる、というのは理解しやすいでしょう。バイアスは、入力データ全体（データの分布）の偏りを調整する機能です。活性化関数の種類が違えば、ニューラルネットワークで出力値の表現が変わるため、入力データとバイアスを活性化関数に入力することで、予測値にあった出力が表現できるようになります。

たとえば、図3-2の例の場合、活性化関数にReLU（ランプ関数）を用いています。ReLUは、入力データにバイアスが加算されたデータが0より小さいデータで

あれば0になり、0より大きいデータはそのままの値で出力される、という非常にシンプルなものです。

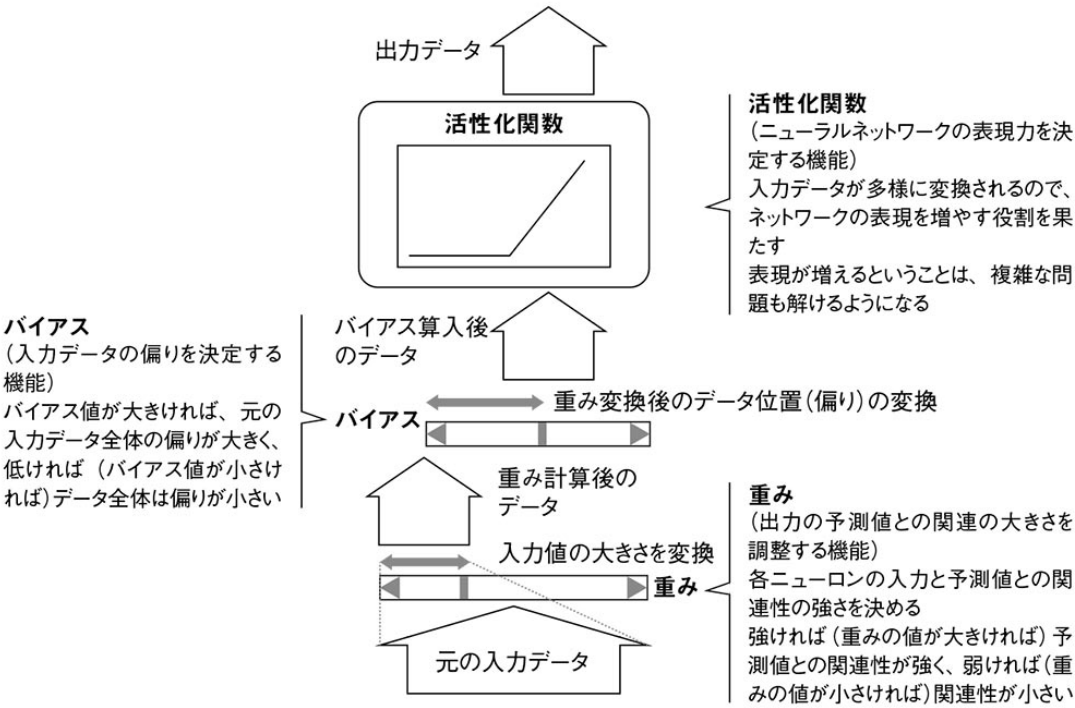


図3-2：入力データが1個の場合のニューラルネットワークの基本機能

■ COLUMN　脳細胞のシナプスとニューロンの伝達の関係

ここでは、ニューロンにおける情報の伝達の仕組みを説明します。ニューロン間の情報伝達をつかさどるシナプスまわりの詳細なイメージとして次の図3-3を参照してください。

ニューロンは通常、何も刺激がない場合にはニューロン内の電位は低い状態にあります。外界から刺激を受けることで一瞬だけその電位が正（活動電位）になった後、再び元の電位に戻るような仕組みが存在します。この電気的な刺激は、発火（スパイク）と呼ばれます。

発火は、外部からの刺激の強さがある一定以上の値（しきい値）になったときに起きます。また、ニューロンは発火した際に、軸索を通して軸索の末端にあるシナプスから神経伝達物質を放出して他のニューロンの電位を変化させることで、刺激をニューロン間で伝搬していく仕組みになっています。

ニューロンがほかのニューロンを発火させたとき、つまり情報を伝達させたときには、その2つのニューロン間の結合が強まります。また、長い間発火しなかったニューロン間の結合は弱まっていきます。このニューロン間の結合の強弱が、人の情報の記憶のメカニズムに関与しています。人はある外界の刺激に対する経験を学習するために、脳全体のニューロン間の結合の強弱をマッピングしていると考えられています。

繰り返し発火するようなニューロン間の結合は強化される一方で、長期にわたり発火が起こらないニューロン間の刺激は減衰する、といった仕組みは「ヘッブの法則」と呼ばれています。

つまり、似た経験が繰り返されると、似たニューロン間の結合が強化されていくことで、その経験は強く記憶されていきます。

人工ニューロンにおいては、この発火のスパイクの形が活性化関数、発火のしきい値がバイアス、ニューロン間の結合の強弱が重みに対応しています。

以上の説明からニューラルネットワークは人の脳の機能を抽象化したものであることがおわかりいただけたのではないでしょうか。

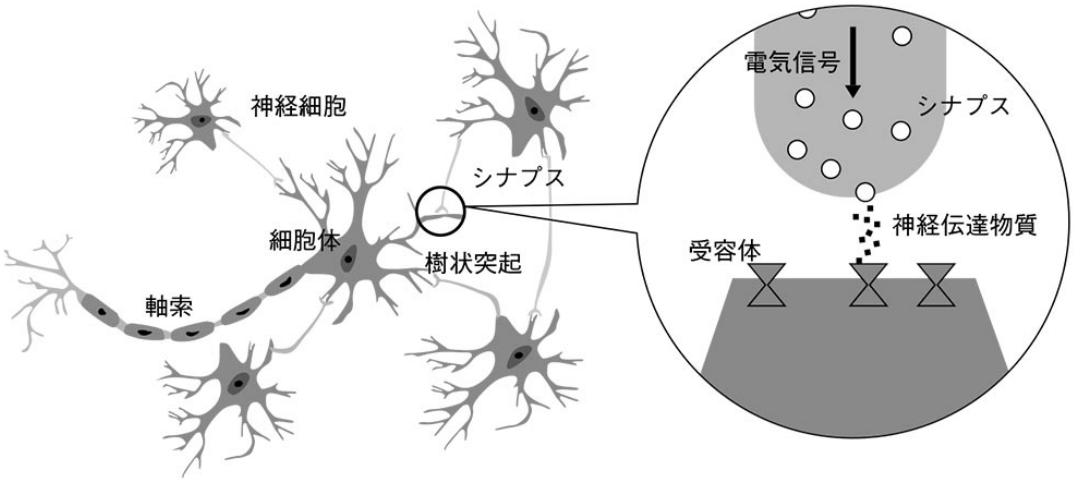


図3-3:シナプスの構造

▶3.1.2 ニューラルネットの可視化 ― 数学的な理解のために

ニューラルネットワークの説明では、以下のような図がよく使用されるため、読者の皆さんもこのような図をよく目にしていることでしょう。

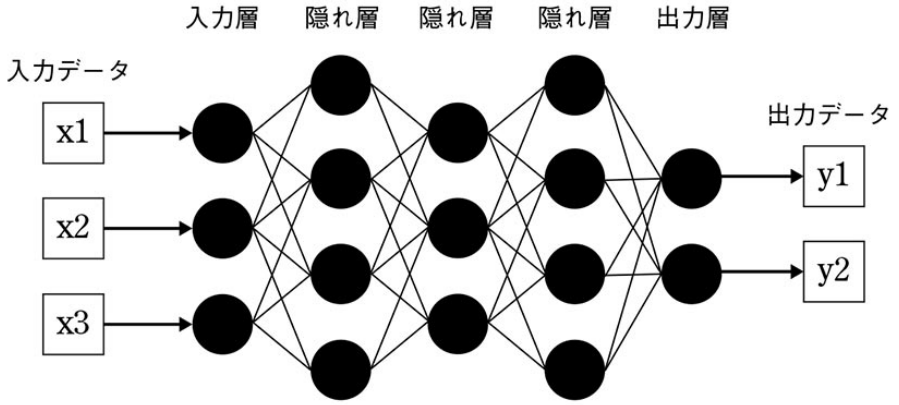


図3-4:ニューラルネットワークの例

図3-4は、ニューラルネットワークの処理の流れを示したものです。この図は、脳科学的な説明という観点でも理解しやすいものですが、数式の処理の流れを可視化するという観点でもよく用いられます。以下では、数式的な側面について説明します。

●計算グラフ(Computational Graph)

数式の表記方法の1つに「計算グラフ」というものがあります。計算グラフとは、計算過程を図3-5のようなグラフを用いて可視化して表現することで、計算の流れをグラフィカルに理解するためのものです。ニューラルネットワークは、図3-4のニューラルネットワークの例のように、入力から出力までのネットワークの計算の流れで構成され、実際の計算の流れを説明する際には計算グラフが用いられます。ここでは、具体的な計算グラフの例を示しながら、基本的な概念を説明します。

図3-5の例は、(6+4)×(4−2)という式について計算グラフを用いて説明したものです。計算グラフは以下の3つの基本的な要素から構成されます。

●リーフ(葉):入出力データなどを含めた変数を表現(図の□)

●ノード:実際の各演算処理を表現(図の○)

●エッジ:処理の進む方向を表現(図の→)

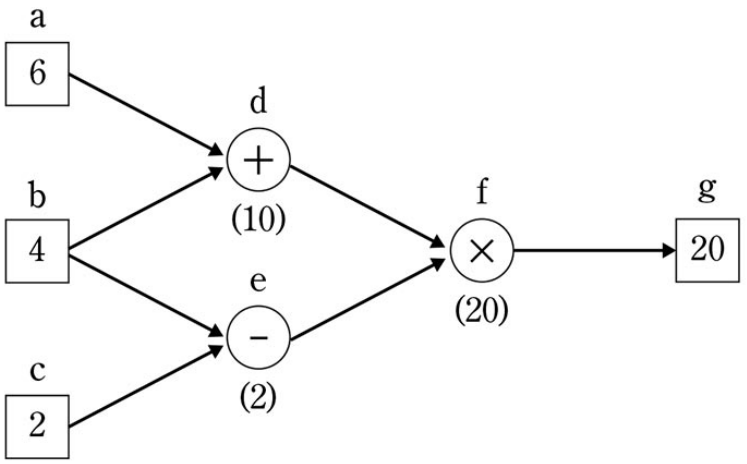


図3-5：計算グラフの例

今回の説明では便宜上、それぞれの各リーフにa、b、c、g、ノードにd～fという名前を割り振ります。さらに実際の計算を追いやすいように、途中の計算結果を丸括弧（）で括って表記します。

まずd=a+bに注目してみます。リーフaとbから入力された数値がdのノードに入り加算される、という仕組みを表現しています。同様に、fに着目するとdとeの入力がfのノードで乗算されることを示しています。計算グラフはこのように、グラフによる可視化機能を利用して数式の処理の流れを表現しています。皆さんの中で関数型プログラミングをお使いの方や、分散システムにおけるデータ処理フローのようなものを活用されていれば、すでに慣れ親しんでいるものかもしれません。ここで示した図3-5は読者の皆さんもお察しのとおり、ニューラルネットワークの視覚的表現そのものとしても利用されています。ニューラルネットワークにおける各ノードの処理では、基本的に各入力値を合計し活性化関数で変換しているので、計算グラフとの親和性が高いのです。そのため、ニューラルネットワークの説明では、計算グラフを用いた表記がよく利用されています。

3.2 ニューラルネットワーク処理プロセスの概要

ニューラルネットワークを実際に利用する場合、一般的に図3-6のプロセスを用

います。このプロセスは基本的にディープラーニングでも変わりません。

例として車の運転を考えてみてください。誰でも無免許でいきなり車を買って公道を走ることはできませんよね。図3-6の下のように教習、仮免、公道走行、それぞれに合格して晴れて免許取得となることで、初めて運転が可能になります。ニューラルネットワークでも事情は同じで、最初の学習フェーズで、モデルの訓練用のデータを使ってニューラルネットワークを学習させ、学習で十分な結果が出たら、訓練データとは別の新たなテストデータを使って学習済みのモデルをテストします。そのテストでも十分な精度が出たら、実環境でテストを行い、そこで合格してようやく商用で使える環境に展開できます。

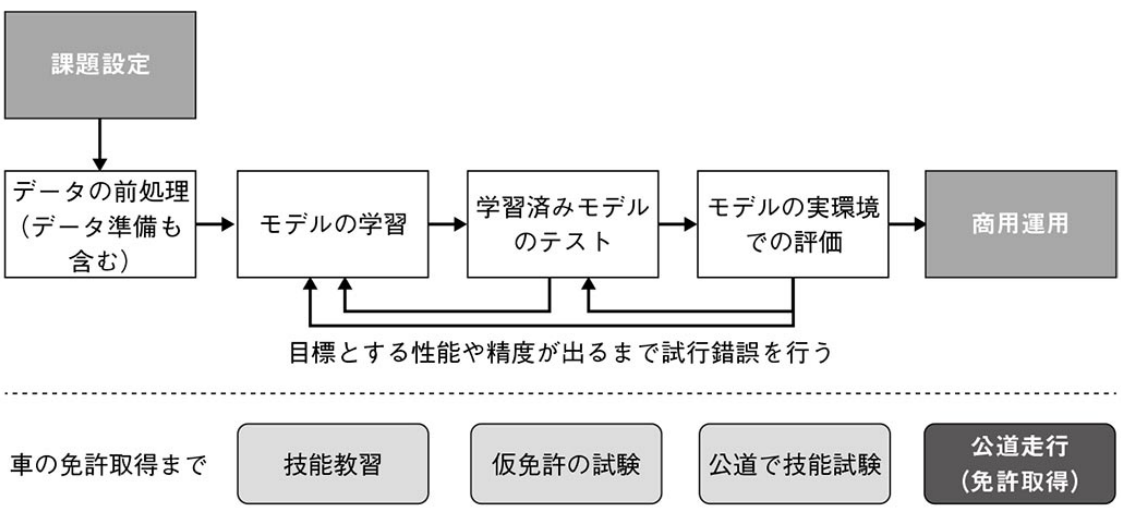


図3-6:ニューラルネットワークの処理プロセスの概要

ニューラルネットワーク処理プロセスのおおまかな内容について理解できたら、次にそれぞれのフェーズで用いられている、学習に関する検証および評価で用いる技術を見ていきます。基本的な技術として、学習フェーズで用いるものと評価フェーズで用いるものには本質的に違いはありません。そのため、以降の説明では、最初に学習と評価フェーズに共通する一般的な技術について説明し、最後にニューラルネットワークの評価の節を設け、評価フェーズを実行する際に特別に必要となる知識について説明していきます。

3.3 ニューラルネットワークの学習

前節までで、ニューラルネットワークの構成の基本機能およびプロセスの概要について説明しました。本節では、ニューラルネットワークの学習のプロセスにおける動作について説明していきます。機械学習やディープラーニング（深層学習）という名称にも含まれているとおり、ディープラーニングでも学習の動作を習得することが肝心です。ディープラーニングを日々扱っていくときに重要なことは、学習で正しい結果を得ていくように試行錯誤を繰り返すことです。現場での試行錯誤が正しい方向で繰り返されているかをチェックしやすくするには、学習の動作について具体的なイメージを持つことが重要です。

ニューラルネットワークに限らず、学習の目的は、予測値に対して正解値をできるだけ正しく出力してくれるような関数を見つけ、未知のデータに対しても正しく数値やパターンを予測することにあります。学習の目的を達成するために、ニューラルネットワークでは、正解値とニューラルネットの出力層から得られる予測値が可能な限り近くなるように、それぞれのニューロンのパラメータである重みとバイアスを調整していきます。なお、ディープラーニングではニューロンを**ノード**もしくは**ユニット**と呼びますが、以降ではノードという呼び方で統一します。

学習は、具体的には図3-7に示すような方法で実現されます。ニューラルネットワークを初めて触れる方でも理解しやすいように、図3-7は教師ありデータを用いたニューラルネットワークの学習の例になっています。教師ありデータとは、入力値に対してあらかじめ正解値が与えられているデータのことを指します。

最初に、ニューラルネットワークの各ノードの重みとバイアスに対して適当な初期値を与えます。それらの初期値と入力データから、予測値を計算します。次に、その計算結果である予測値と、正解値との間の誤差を計算するのに、**損失関数**という指標を用います。一般的に重みとバイアスの初期値を用いて損失を計算する場合、初期値はチューニング前の適当な値であるため、予測値と正解値の誤差は非常に大きなものになります。ニューラルネットワークの学習の目標は、その誤

差を最小にするために重みとバイアスの値を調整していくことです。重みとバイアスの値をやみくもに更新したところで、いつまでたっても誤差が最小の値にならないことは容易に想像がつくでしょう。そのため、ニューラルネットワークでは、**最適化**と呼ばれる手法を用いて、誤差が最小値にたどり着くまで重みとバイアスの更新を続けていきます。損失関数や最適化の手法については、のちほど詳しく説明します。

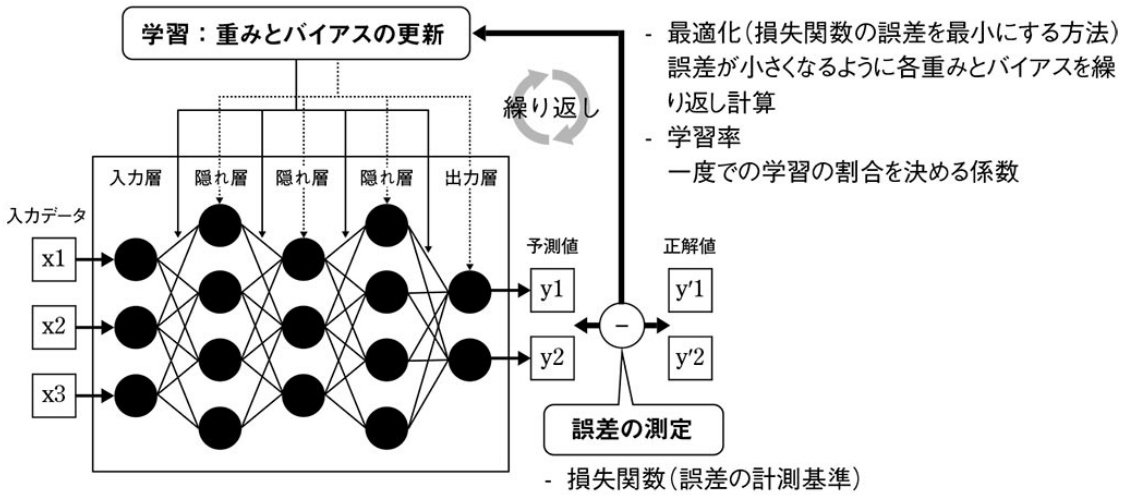


図3-7：ニューラルネットワークにおける学習動作

以上がニューラルネットワークの学習の概要となります。次節以降では、具体的な学習の動作を見ていくための下準備と、実際の細かい学習の動作を説明します。これらは本章のメインテーマなので、しっかり学んでいきましょう。

3.4 ニューラルネットワークの各機能

前節では、ニューラルネットワークの学習の仕組みの概要を見てきました。本節では、具体的な学習の動作を理解するための下準備として、ニューラルネットワークを構成する各構成要素の機能についておさらいも兼ねて説明していきます。

▶3.4.1 重みとは

重みは、各ノードの間をつなぐネットワークの結合の強さを示すパラメータです。結合の強度が強いほど前段のノードの情報を伝えやすく、弱いほど伝えにくくなります。情報が伝えやすいということはすなわち、その前段のノードの情報が出力結果に対して果たす役割が大きいこと、言い換えると出力結果との関連性が高いことを意味します。逆に、情報が伝えにくいとは、出力結果に対して前段のノードの情報の関連性が低いことを意味します。それぞれの重みは、ある入力に対してはより強く反応し、別の入力に対してはより弱く反応するように作用することで、後段のノードにそれぞれの情報を伝えていきます。ニューラルネットワークにおける学習とは、全入力データに対する予測値と正解値の誤差が一番小さくなるようにすべての重みの大きさを調整していくことなので、重みの調整が学習のキーファクターになります。

▶3.4.2 バイアスとは

バイアスは、データの特徴に従った偏りを調整する役割を果たすものです。もともとの入力データのデータ分布に、ある特徴に従った偏りがある場合には、たとえば、あらかじめその偏りのデータ分布の中心値をもとにデータを調べられるようにしておくことで、特徴をとらえやすくなります。イメージとしては、男性の特徴をとらえたい場合、身長の特徴として170センチを中心にデータを調べたほうが、160センチを中心に調べるよりも効率的に特徴をとらえることが可能になるということです。

特徴をとらえやすくなるということは、予測値に対して入力の特徴がオリジナルのデータのものより近い状態になる、つまりはシナプスで考えるところの発火がしやすい状態になったことを意味します。そのため、実際のニューラルネットワークでは、できるだけ予測値に合うように入力データのデータ分布を移動させて調整することが、脳細胞でいうところのニューロンが活性化しやすくなるようにしきい値を調整するのと同一コンセプトとしてとらえられます。

さらに言い換えるとバイアスは、発火のしきい値の高さを調整する役割を果たすとも言えます。バイアスの値が大きい場合にはしきい値との差が小さくなるため、発火がしやすくなるのに対し、バイアスの値が小さい場合にはしきい値との差が大きくなるため、発火しにくくなるように作用します。バイアスは発火の判定基準としてのノードの感度であると言い換えることができます。

このような観点でとらえた場合、重みが前段のノードによる出力の伝達のしやすさを制御する役割を果たしているのに対し、バイアスはノードの発火のしやすさを制御する役割を果たします。どちらも情報の伝達のしやすさを制御するために機能するという点では同じですが、それぞれの持つ役割は異なることを理解してください。

実際のバイアスの調整は、重みとまったく同じ手法で実現可能です。重みは、前段のノードの出力結果に重みを掛け合わせた値を次のノードの入力として用います。対して、バイアスは、前段のノードからの出力がなくバイアスの値が直接次のノードの入力になっている点が異なります。ただし、このことはある意味、図3-8にあるように、バイアスの場合には前段のノードが1という値に固定されているものだと考えることもできます。そうすることで、バイアスは前段の出力結果を掛け合わせることを含めて、重みの動作の一種とみなして処理できるようになります。そのため、実際の計算ではバイアスも重みと特別な区別をせずに計算できます。

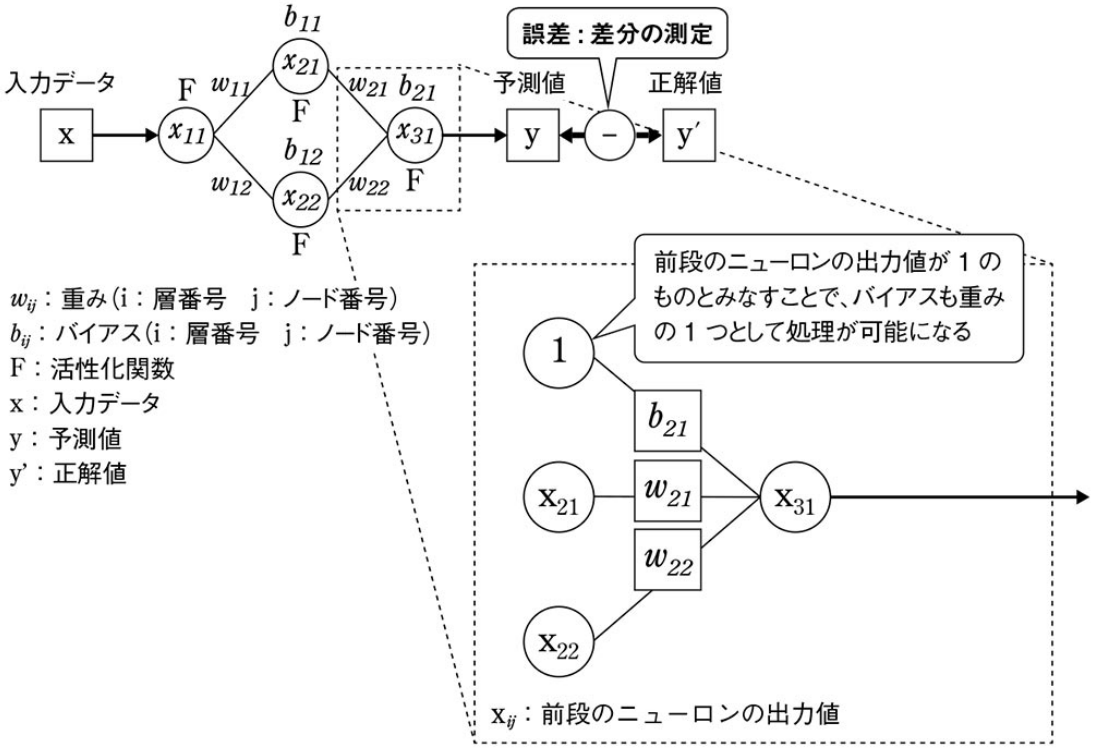


図3-8：バイアスを重みの一種として計算することが可能となる理由

▶3.4.3 活性化関数とは

活性化関数とは、ノードに集まった入力信号の総和をどのように活性化するかを決定する関数です。

活性化関数は大きく分けて、2種類存在します。隠れ層で用いられる活性化関数と、出力層で用いられる活性化関数です。さらに、出力層の活性化関数は、解決したい問題が分類か回帰かによって、2種類に分かれます。以降では、まず隠れ層で用いられる活性化関数について説明し、その後で出力層の活性化関数を説明します。

▶3.4.3.1 隠れ層の活性化関数

隠れ層の活性化関数は、入力データから正解値を正しく表現するために必要な特徴を学習する機能を持ちます。対して、出力層における活性化関数は、出力データである予測値がどれだけ正しく正解値をとらえているかを調整するために

用いられます。

つまり、隠れ層における学習とは、隠れ層の前の層の全入力データに対して、隠れ層の特徴の組み合わせのうち予測値に対する正解値に一番近くなるものを見つけ出す作業となります。ちなみに、この特徴の組み合わせは一般的に**特徴量マップ**と呼ばれます。

隠れ層の活性化関数は、入力データから出力値までの間で、非線形の関係性を学習するものです。隠れ層の活性化関数として以前はシグモイド関数が使用されていましたが、現在はReLU関数が一般的に用いられます。なお、シグモイド関数については、本項の出力層で用いられる活性化関数のところ（「3.4.3.3 分類問題の出力層で使う活性化関数」）で説明します。

◉―― ReLU 関数（ランプ関数）

ReLU関数は、図3-9のとおり、0以下の入力値に対しては0を出力し、0以上の入力値に対しては、入力値をそのまま出力する関数です。

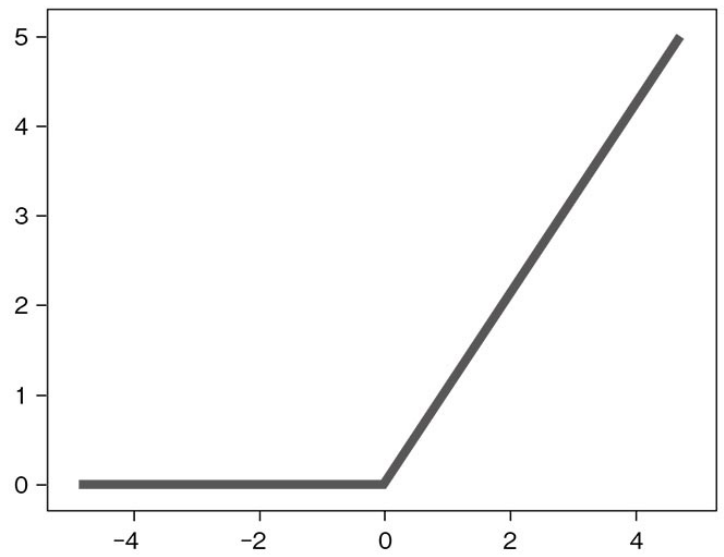


図3-9:ReLU関数

■COLUMN　活性化関数に線形関数が用いられない理由

活性化関数に線形関数を用いない理由は、ディープニューラルネットで線形関数を用いて層を深くしても同じ変換を繰り返すだけになってしまうからであり、また、隠れ層がないネットワークを用いても同様の変換が実現できてしまい、層を深くした意味がないからです。

非常に簡単な例としては、$y=2x$が活性化関数とすると、2層目は$y=(2(2x))=4x$、3層なら$y=2(2(2x))=8x$というように、3層で$y=2x$の変換を施したものが、$y=8x$を用いれば一層で表現できてしまうので、隠れ層である必要性がなくなり、多層にする利点がなくなってしまいます。そのため、隠れ層では線形関数は使うべきではありません。

▶3.4.3.2　回帰問題の出力層で使う活性化関数［恒等関数］

図3-10からわかるように、**恒等関数**は入力値をそのまま出力する関数です。出力層の回帰問題では、予測された数値をそのまま出力して利用することが多いため、一般的に使用されます。

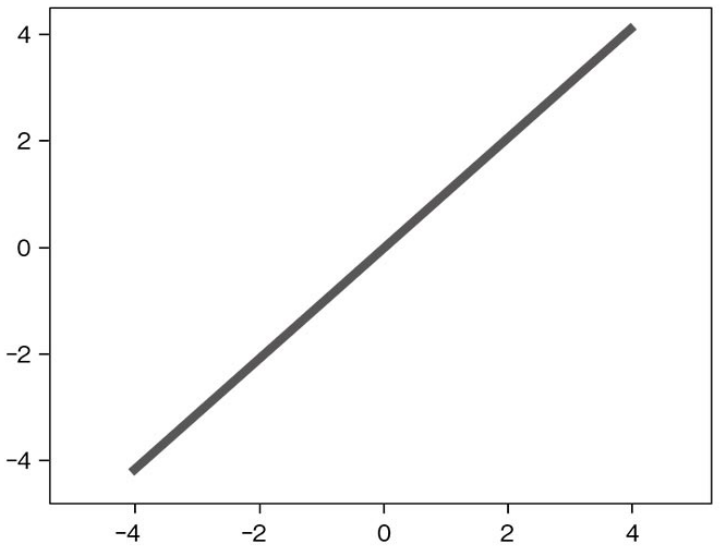


図3-10: 恒等関数

▶3.4.3.3　分類問題の出力層で使う活性化関数［シグモイド関数、ソフトマックス関数］

◉――シグモイド関数

図3-11に示したのがシグモイド関数です。**シグモイド関数**は、入力値を0～1の間の出力値に変換する関数です。負の値が大きい場合には出力値は0に近づき、正の値が大きい場合には1に近づきます。シグモイド関数は二値分類の問題を解く場合に一般的に用いられます。シグモイド関数では入力値が−∞（マイナス無限大）の場合、出力値が0となるため、二値の分類問題に当てはめた場合、たとえば「1となる確率が0%」と読み替えられます。同様に、入力値が0の場合、出力値が0.5となるため、「0となる確率が50%、1となる確率が50%」と表現できます。さらに、+∞（プラス無限大）の場合には、出力値が1なので、「1となる確率が100%」と表現できます。シグモイド関数は以上の理由から、二値のいずれかをとりうる確率として考えることができ、二値を分類する場合に頻繁に用いられています。

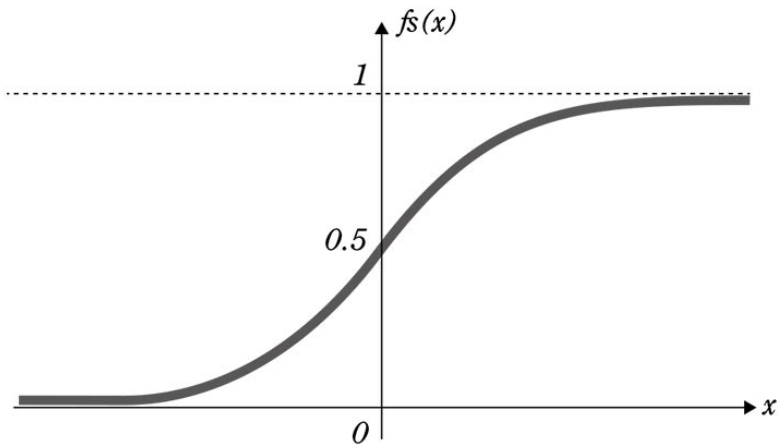


図3-11:シグモイド関数

◉――ソフトマックス関数

ソフトマックス関数は、以下のような数式で表されます（数式3-1）。Σやe（指数関数）が組み合わされているので少し難しく見えるかもしれませんが、複数の入力値のk番目の要素を指数変換し、さらに全入力値を指数変換したうえで、各要

素の指数表現を全入力値の指数表現の和で割っているだけです。ソフトマックス関数の特徴は、以下のとおりです。

- それぞれの出力要素はすべて0～1の範囲におさまる
- すべての出力値の合計が1になる
- 入力値の中で一番大きい値が出力値では指数表現になることでより強調され、より支配的な要素となる

$$S(y_k) = \frac{e_k^y}{\sum_{k=1}^{n} e_k^y}$$

S：活性化関数（ソフトマックス）　　y_k：k番目の予測値
n：入力データの全数　　e_k^y：k番目の予測値の指数関数
k：入力データの番号　　$\sum$：総和記号

数式3-1：ソフトマックス関数

ここでソフトマックス関数についてより詳細にイメージしてもらうために、具体的な利用例を示しましょう。

図3-12は、ひらがなの崩し字の画像データの文字を入力とし、入力画像がひらがなのうちどれにあたるかを予測するモデルにディープラーニングを用いた例です。この例では、ディープラーニングのモデルの出力結果としての特徴量を入力とし、ソフトマックス関数を活性化関数として用いて、それぞれの入力値を変換することにより、入力画像が“あ”から“ん”のひらがなのラベルのうちどれにあたるかを確率値として出力できるようになります。つまり、ソフトマックス関数は、多クラス分類の問題として利用可能な活性化関数であることを示しています。

この図では、崩し字の「あ」という文字を入力した場合、その出力層の結果を右の確率の値に変換するためにソフトマックス関数を用いています。出力値がいかなる値であっても、ソフトマックス関数を用いることで出力の合計値が1となるため、各出力値は各出力ラベルの確率として使用できるようになります。以下の例で

は、「あ」というラベルの数値が0.75なので「あ」というラベルである確率が75%、「い」というラベルである確率が3%というように、各出力ラベルになる確率がどのくらいあるかを示す値として使用できることを意味します。

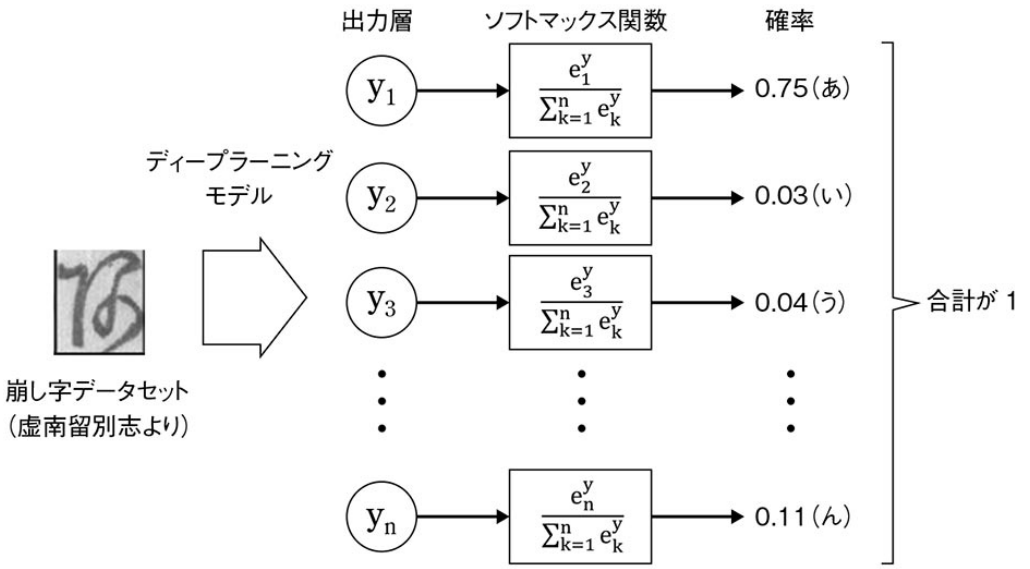


図3-12：ディープラーニングモデルの出力層の操作イメージ

▶3.4.4 損失関数とは

ニューラルネットワークの学習の中で損失関数の果たす役割は、正解値と予測値の誤差を計算することです。このことは、損失関数の出力値が学習精度の測定基準になることを意味します。この出力値が大きい場合にはモデルの精度が悪く、小さい場合にはモデルの精度が良いと判断できるようになるため、損失関数の出力値が最小になるようにパラメータを調整することがニューラルネットワークの学習の最終目標になります。

分析すべき課題には一般的に、回帰と分類の2種類があります。ディープラーニングにおける損失関数も同じく、分析すべき課題が回帰か分類かの違いによって2種類存在します。そのため、最初に回帰と分類の違いについて簡単な説明を行ってから、それぞれの課題で選択すべき損失関数がどのように違うのかを説明します。

図3-13では、回帰と分類の違いを示しています。

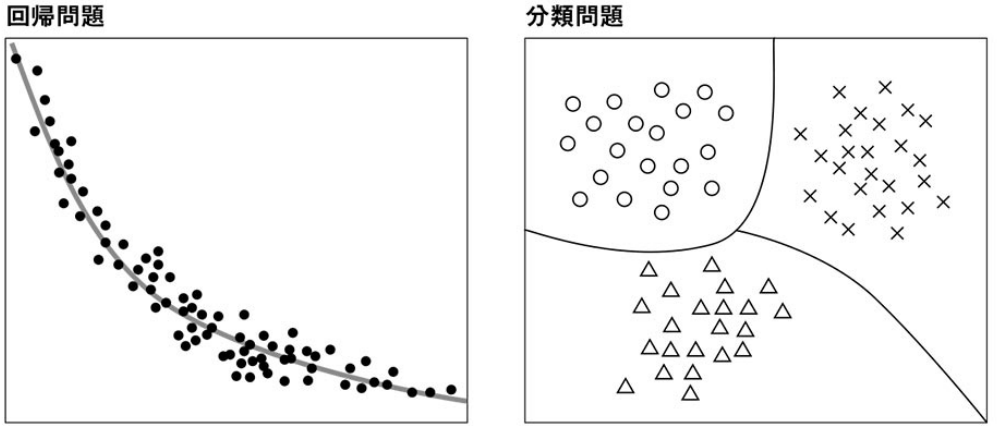


図3-13: 回帰と分類の違い

この図の左が回帰問題であり、連続値の予測を行います。回帰で扱う入力データはもっぱら数値データとなります。

図3-13の右が分類問題であり、入力データがどのクラスに属するかを予測します。図の例では○×△の3つの分類ですが、分類の中でも特に2つのクラスに分類することは二値分類と呼ばれ、3つ以上の分類は多クラス分類と呼ばれます。分類で扱う入力データはカテゴリデータや数値データですが、連続値ではなく離散値を扱うケースが一般的です。

▶3.4.4.1 回帰問題で用いられる損失関数

回帰問題では、正解値と予測値の誤差は実際の距離の違いとして測定可能です。

ニューラルネットおよびディープラーニングでの学習では訓練用データが多量に必要なので（ビッグデータであることが望まれる）、予測値と正解値の誤差も1つのサンプルデータではなく、訓練用の全データに対する誤差の合計として求める必要があります。そのため、損失関数としてすぐに思いつくのは、訓練用の全データに対して予測値と正解値の誤差の絶対値をとったものを足し合わせたのち平均化するために、訓練データの全数で割るというものです。この数式は**平均絶対誤差**と呼ばれます。数式のほうが理解しやすい方のために以下に数式を示し

ておきます。

$$E = \frac{1}{n}\sum_{k}|y_k - \hat{y}_k|$$

E：損失関数（平均絶対誤差）　　$\hat{y}_k$：k番目の予測値

n：データの全数　　y_k：k番目の正解値

k：データの番号

数式3-2:平均絶対誤差

単純に損失関数の誤差を計算すると、誤差が正の値になることも負の値になることもあります。たとえば、あるサンプルデータでは10という正の誤差があり、別のサンプルデータでは−10という負の誤差があったとします。その場合、それを単純に足し合わせてしまうと、誤差が相殺されてゼロになってしまいます。このような状況を回避するために、平均絶対誤差では、絶対値にすることで、必ず誤差が正の値になるように工夫しています。

ディープラーニングの回帰問題で最もよく用いられる別の方法は、損失関数の誤差評価に**平均二乗誤差（Mean Squared Error）**を使うものです。平均二乗誤差では誤差の値を絶対値の代わりに2乗することで、すべての値が確実に正の値になるようにしています。平均二乗誤差の実際の数式は以下のとおりです。

$$E = \frac{1}{n}\sum_{k}(y_k - \hat{y}_k)^2$$

E：損失関数（平均二乗誤差）　　$\hat{y}_k$：k番目の予測値

n：データの全数　　y_k：k番目の正解値

k：データの番号

数式3-3:平均二乗誤差（MSE［Mean Squared Error］）

▶3.4.4.2　分類問題の損失関数

分類問題は特にカテゴリデータの場合、正解値と予測値との実際の距離には

まったく意味がありません。その代わり、分類問題では、正解値の集合の確率分布と予測値の集合の確率分布の違いを誤差として換算するほうが正しく評価できます。この確率分布間の誤差を計測する指標として適した損失関数が、下の数式3-4で示される**交差エントロピー**です。

$$E = -\sum \hat{y}_k (log\, y_k)$$

E：損失関数（交差エントロピー）　　y_k：k 番目の正解値
n：データの全数　　$\sum$：総和記号
k：データの番号　　log：対数記号
$\hat{y}_k$：k 番目の予測値

数式3-4：交差エントロピー

この式は少しわかりづらいので、さらに詳しく説明します。

予測値が持っている分類結果の確率分布と、正解値が持っている分類結果の確率分布が同じである場合、予測に基づく確率分布をもとに計算した出力結果は正解値と同程度の確からしさになります。

このことを具体的に説明しましょう。次の図3-14のような正解値の箱と予測値の箱があり、白の玉と黒の玉が入っているとします。正解値および予測値の箱のそれぞれの玉を引く確率が同じである場合、正解値の箱と予測値の箱から1つずつ玉を引いた場合に、背後にあるそれぞれの確率分布が一致しているので、引いた玉の色が同じになる確率が一番高いわけです。

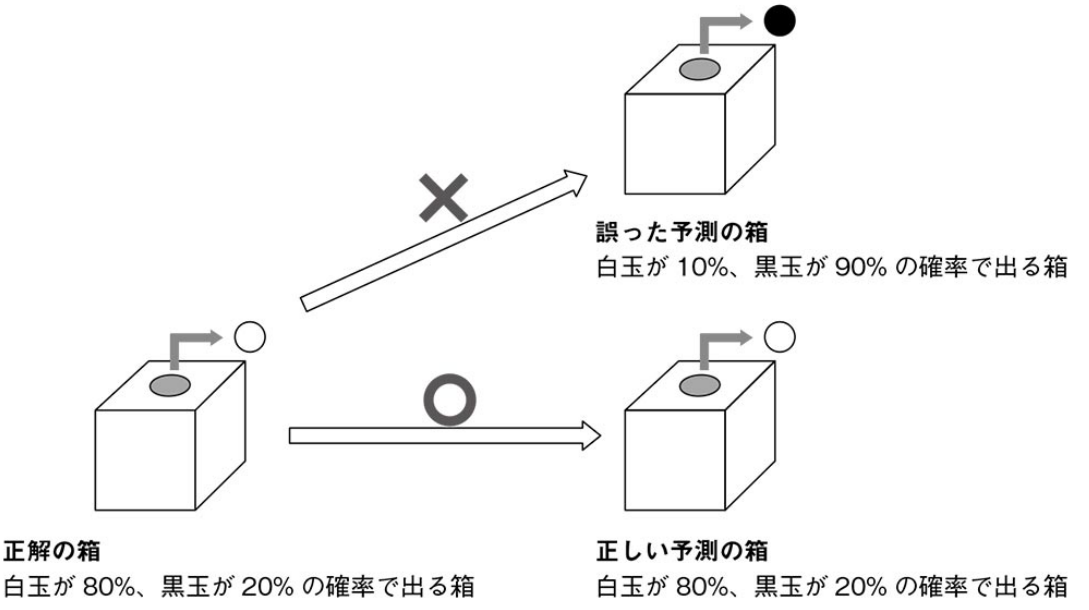


図3-14:正解値と予測値の確率分布の関係

実際、両方の箱に等しく白の玉8個と黒の玉2個が入った場合（80%の確率で白玉が取り出せる箱）と、予測値の箱には白の玉が1個、黒の玉が9個入った箱（10%の確率で白玉が取り出せる箱）を用意した場合では、前者のほうが正解値と同じ玉を引く確率が断然高いわけです。

ソフトマックスやシグモイドは、分類問題の出力層で使用され、出力が確率分布となる活性化関数です。そうした活性化関数と組み合わされる損失関数として最も適切なのは、確率分布の誤差を測定可能な交差エントロピーなのです。

なお、交差エントロピーの数式を細部にわたり説明するには、数式を読み解く必要がありますが、その説明は本書の目的ではないため割愛します。詳しく知りたい方は、アダム・ギブソン氏の共著者『詳説Deep Learning ―実務者のためのアプローチ』を参照してください。ちなみに、ディープラーニングで交差エントロピーが利用される実質的なメリットは以下のとおりです。

- 交差エントロピーは正解値と予測値の誤差が大きい場合、学習スピードが速い
- ソフトマックス関数と交差エントロピーを合成した関数の傾き（勾配）を簡単かつ直感的にわかりやすい式で解くことが可能

以上が代表的な損失関数の説明です。損失関数にはここで紹介した以外にもさまざまなものが存在します。さらに詳しく知りたい方は、アダム・ギブソン氏の共著者『詳説Deep Learning ―実務者のためのアプローチ』を参照してください。

最後に、参考までに本章で扱った主な損失関数と活性化関数の組み合わせを表形式で整理しておきます。

層の種類	活性化関数
出力層	恒等関数
隠れ層	ReLU 関数

層の種類	解くべき問題の種類		損失関数	活性化関数
出力層	回帰		最小二乗誤差	恒等関数
	分類	二値	交差エントロピー	シグモイド関数
		多クラス	交差エントロピー	ソフトマックス関数

表3-1：損失関数と活性化関数の組み合わせの表

本節の説明は以上です。ここまでで、ニューラルネットを形成する基本的な構成要素である、重み、バイアス、活性化関数、損失関数について理解できたはずです。それではいよいよニューラルネットワークのメインテーマである学習について見ていきましょう。

3.5 ニューラルネットワークの学習の動作

ニューラルネットワークの学習では、これから説明する最適化を行う必要があり、その実現方法として勾配降下法が用いられます。これらの動作を理解しておくことは、実際の現場でニューラルネットワークの学習が進まなくなった際に、最適化のどのあたりに問題がありそうかを探るのに役立ちます。

▶3.5.1 最適化とは

最適化を例えると、図3-15のような、山から一番低い場所に楽に速く下りていく

方法を見つけることに似ています。

出発地点が、正解値と現時点の予測値との間の誤差だとすると、最適化とはその間の誤差が最小となるように調整していくことです。予測値と正解値の間の誤差を図3-15のように高さに見立てると、誤差の最小化は、山の高さが一番小さくなる図3-15の山の一番低い場所にたどり着くことと読み替えることができます。

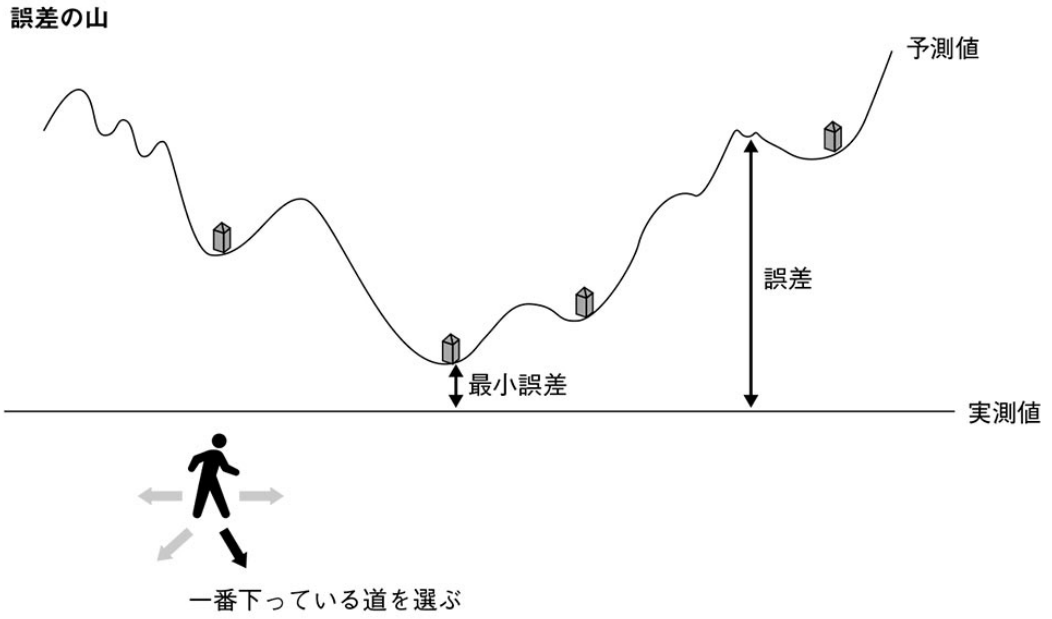


図3-15:誤差の山で一番低い場所（最小誤差）を見つけるイメージ

さて、どのようにして山の中から一番低い場所にたどり着くのが賢いと思いますか?

全部の場所の標高を調べて最も低い場所を見つけることもできますが、それでは非効率極まりないですよね。たとえば、すり鉢状の地形とわかっているなら、任意の場所からボールを転がして一定時間後にボールの動きが止まった場所が一番低い場所になるのは容易に想像できます。

その発想をより一般化して考えると、一番効率が良いのは、今いる場所から一番勾配がきつい下り道を一番低い場所に向かって延々と下っていくことだと理解できます。これを実現する方法が、今から説明する**勾配降下法**もしくは**最急降下法**

と呼ばれるものです。本書では以降、勾配降下法という用語で統一します。

▶3.5.2 勾配降下法とは

実際に山を下りるプロセスでも少しずつ下りながら、下った先でさらに最も急な下り道を探すことになります。さらに下りの道が見つかったら同じことを繰り返し、これ以上下りる場所がなくなる、つまり周囲で一番低い場所に到達するまで同様の作業を行います。山登りの標高にあたるものは、ニューラルネットワークの例では損失と読み替えられます。その損失の誤差を計測するための大きさを表す指標が、先ほど説明したニューラルネットワークにおける損失関数になります。山で迷わず正しい道を行っているかをチェックするにはコンパスを読んでいく必要があり、ニューラルネットワークでは、そのコンパスにあたる機能が損失関数になります。損失関数の値が正しく最小値に向かっているかを調べる計算方法が、勾配降下法ということになります。

基本的に、山道を移動した結果、前の測定した場所よりも今の場所のほうが下っていたら正しい道を選択したと考えられるのと同じく、損失関数の計算結果がパラメータ更新前の損失関数の大きさに比べて小さくなっていれば、基本的に一番低い場所に向かって正しく下りてきていると判断できます。その中でも最も効率良く下るためには、一番勾配がきつい道を選び続けて一番低い場所へたどり着くことが求められます。この一番勾配がきつい道を選択し続けるための手法が勾配降下法です。

これから、勾配降下法を用いたニューラルネットワークの学習プロセスである重みとバイアスの調整について説明します。

補足になりますが、先ほどのバイアスの説明でも触れたとおり、実際の計算の場面では、バイアスも重みの一種と考えて支障ありません。そのため、以降の説明では、損失関数の誤差が最小となるまでの重みとバイアスを更新していく手法について、バイアスを重みの一種とみなして重みの調整に焦点を当てて説明していきます。

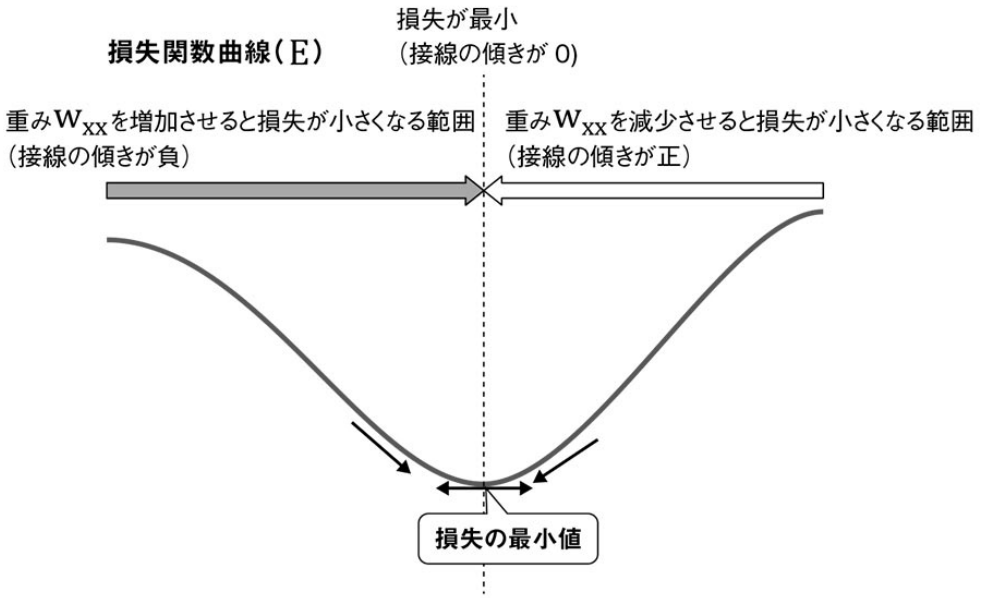


図3-16：損失関数における最小値の求め方のイメージ

図3-16を見てわかるとおり、損失を最小にするには、現時点の損失が図の曲線の損失の最小値より左側にある場合であれば、それぞれの時点の接線の傾きが下る方向に移動していき、最小値のところではその傾きが0となるまで右側に移動していくことで実現できます。逆に最小値よりも右側にある場合、傾きを負の方向つまり図の左方向に移動していきます。つまり、図のような非常にシンプルな起伏の曲線の最小値を見つける場合には、現地点の曲線の傾きを計算してその傾きが0になるまで損失の値が小さくなる方向に移動して調整していけばよいのです。勾配降下法と呼ばれる理由は、傾きを計算し下る方向に調整していくためです。図3-17では、今までの説明を具体的な数式にしています。

ただし、勾配降下法をそのまま何も考えずに実装してしまうと、ディープラーニングの場合、学習に用いるデータやパラメータの量が非常に多いため、計算時間がかかりすぎて実用に耐えられないケースがあります。そのようなケースでも許容される時間内で処理を終わらせるために、勾配降下法のコンセプトに従いながら効率的な計算を実現するいくつかの手法が存在します。ディープラーニングでは、ミニバッチ確率的勾配降下法が最もよく用いられます。以下では参考のために、ミニバッチ確率的勾配降下法とよく比較される確率的勾配降下法とバッチ勾配

降下法についても説明します。

重み更新の公式

$$w_{t+1} = w_t - \gamma \frac{\partial E}{\partial w_t}$$

E ：損失関数
w_t ：更新前の重み
w_{t+1}：更新後の重み
γ ：学習率
∂ ：微分記号（偏微分）

重みw_tが損失の最小値よりも左側にある場合：

曲線の接線の傾き $\frac{\partial E}{\partial w_t} < 0$ ⟹ 重み w_{t+1} を増やす
（重みw_{t+1}を右側に移動する）

重みw_tが損失の最小値よりも右側にある場合：

曲線の接線の傾き $\frac{\partial E}{\partial w_t} > 0$ ⟹ 重み w_{t+1} を減らす
（重みw_{t+1}を左側に移動する）

図3-17:重みの更新の概要

▶3.5.2.1 確率的勾配降下法

確率的勾配降下法では、全データから毎回1つのサンプルデータを抽出し損失関数の誤差を計算してパラメータを更新していきます。のちほど説明するバッチ勾配降下法と比べると精度は良くありませんが、増えた分だけの学習データを使い、重みの初期値は前回の学習結果をそのまま用いて再学習できるので、再学習の計算コストが劇的に小さくなります。

確率的なので、運が良ければ勾配降下法よりも早く最適解にたどり着けますが、運が悪ければいつまで経っても最適解にたどり着くことができません。確率的勾配降下法では、1つのサンプルデータが入ってくるたびに学習を行う仕組みとなっています。

▶3.5.2.2 ミニバッチ確率的勾配降下法

ミニバッチ確率的勾配降下法は、ディープラーニングで一番よく用いられる勾配降下法です。確率的勾配降下法ではパラメータの更新を1つのサンプルデータごとに行っていましたが、ミニバッチ確率的勾配降下法では、学習データの中からランダムにいくつかのデータをひとまとまり（**バッチサイズ**と呼ばれます）にして取

り出してから誤差を計算し、パラメータを更新していきます。確率的勾配降下法と、この後で説明する**バッチ勾配降下法**との中間に位置するようなものです。確率的勾配降下法では、学習結果が1つ1つのデータに含まれる固有のノイズに大きく反映されてしまうという欠点があり、バッチ勾配降下法では学習に時間がかかるという欠点があります。ミニバッチ確率的勾配降下法では、バッチサイズごとにデータを取り出して誤差を計算しパラメータを更新するため、個別のノイズによる不安定さを抑えつつ、バッチ勾配降下法より計算時間を短縮できるというメリットが得られます。

▶3.5.2.3　バッチ勾配降下法

バッチ勾配降下法になぜ確率という名前がないかというと、確率的勾配降下法では全データの中から一部のデータをランダムに選択するのに対し、バッチ勾配降下法では全データを用いて計算するからです。バッチ勾配降下法のデメリットとして、学習データが多いと計算コストが非常に大きくなってしまうことが挙げられます。また、学習データが増えるたびにすべての学習データで再学習が必要となってしまい、計算に非常に時間がかかってしまうというデメリットもあるため、現場ではあまり使われません。

代表的な3つの勾配降下法についての説明は以上となりますが、まずミニバッチ確率的勾配降下法の考え方をしっかり押さえて、現場で試すようにしてください。

次にそれぞれの降下法の違いをまとめましたので参考にしてください。

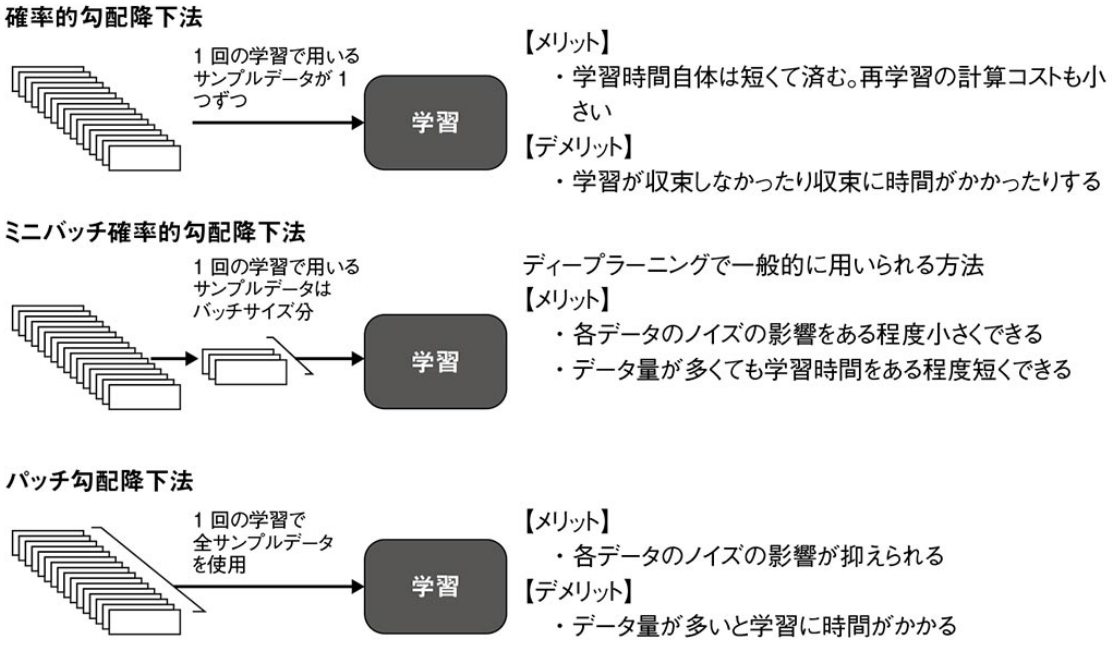


図3-18:各勾配降下法のまとめ

▶3.5.3 局所解の問題

勾配降下法を用いれば基本的に簡単に最小値が見つけられそうですが、実際に一番低くなる場所を探すのは難しいものです。理由は、あらかじめ一番低い場所がわかっていないうえに、実際の山でもそうですが、複数の谷が存在していて自分では一番低い場所にたどり着いたつもりでも、本当の意味で山全体を見渡した場合には、一番低い場所ではない可能性があるからです。実はある場所では周りを見渡すと確かに一定の一番低い場所なのですが、全体で見た場合には別の場所が一番低いということがよくあります。ちなみに、このように見渡せる一定の範囲内における一番低い場所は、ディープラーニングなどでは**局所解**と呼ばれます。ディープラーニングの学習では、山から下って行くときに一番低い場所まで下りきったという保証がありませんし、さらには山の出発地点もあらかじめ決まっているわけでもありません。ちなみに、山全体で一番低い場所は**最適解**と呼ばれます。そのような状況でもある程度の時間内にある程度の確からしさで損失誤差の最小値を見つける必要があるため、先のセクションで紹介したシンプルで基本的な勾配降下法を用いた最適化のほかに、さまざまな最適化方法が提案され

ています。これらの最適化方法も日進月歩で多様になってきているため、具体的な方法を知りたい方は、アダム・ギブソン氏の共著者『詳説Deep Learning ―実務者のためのアプローチ』を参照してください。

次の節に進む前に、ディープラーニングで数式を用いる際、対数を使う理由を質問される方が多いので、参考までにその理由について簡単に説明します。

■COLUMN　さまざまなものに対数を使う理由について

ディープラーニングに限らず統計や機械学習では、対数を目にすることが多くなります。読者の中には数式にアレルギーがある方もいらっしゃるでしょう。普通の数式でも辛いのに対数まで出てくると理解できないと思われるかもしれませんが、対数で扱うほうが便利なので使っているという理由があります。その便利さについて簡単に説明します。

対数を用いることには、以下のメリットがあります。

- ●対数をとることで、掛け算が足し算に、累乗が掛け算になるため、対数をとる前の式に比べて計算が簡単になる
- ●もともと大きい値だったものを小さい値で表現できる

またディープラーニング的にはおまけの部分になりますが、以下の理由でも対数が用いられます。

- ●人間の多くの感度は対数に合っているため、感度を表す場合に使用されることがよくある

たとえば、地震の震度、音の大きさなど、人が体感する一部のものは実数ではなく対数のほうが感覚的にとらえやすいため、対数を用いて表現されることがあります。

3.6 ニューラルネットワークの隠れ層の学習についての直感的理解

ディープラーニングにおいて学習は非常に重要なテーマです。今からニューラルネットワークの隠れ層の学習について、より具体的で直感的に理解できるような説明をしていきます。読者の方々がいつかこのイメージを持っていてよかったと思えるような瞬間に本書を通して出会っていただけたらうれしいです。

ここでは図3-19の左のような具体的なデータを用いて、ニューラルネットワークの内部の動きを見ていきます。まず図のような、濃いグレーと淡いグレーの点の集合があるとします（波線の境界の上が濃いグレー、下が淡いグレー）。ニューラルネットワークを用いて左の入力データからモデルを作成した場合、それぞれの位置のデータが正しく濃いグレーと淡いグレーの点を分類できるようにすることを目的とします。図3-19の左が正解値のマップで、右がニューラルネットワークを使って学習を行った予測結果となります。濃いグレーと淡いグレーの境界部分にある波線の波の数は予測のほうが少ないのですが、実際の予測の精度は95%を超えるものにまでチューニングできます。

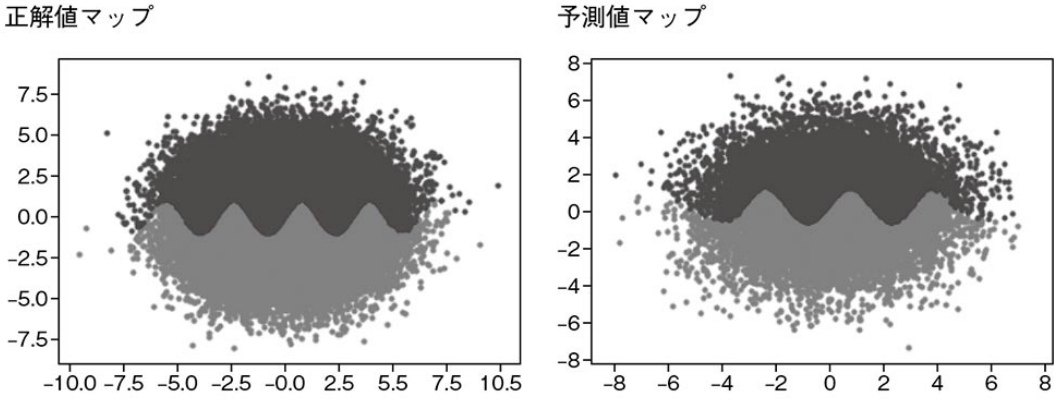


図3-19:例のサンプル入力データとディープラーニング学習後の予測データ

学習用入力データのデータセットは、教師ありデータセットで、それぞれの点の位置情報にあたるx軸、y軸の2つの座標の値と、その位置の点が濃いグレーか

淡いグレーかのラベル情報になります。なお、今回の学習で用いたニューラルネットワークの構成は下の図3-20のとおりです。隠れ層は3層で、それぞれの層は32ノードで構成されています。

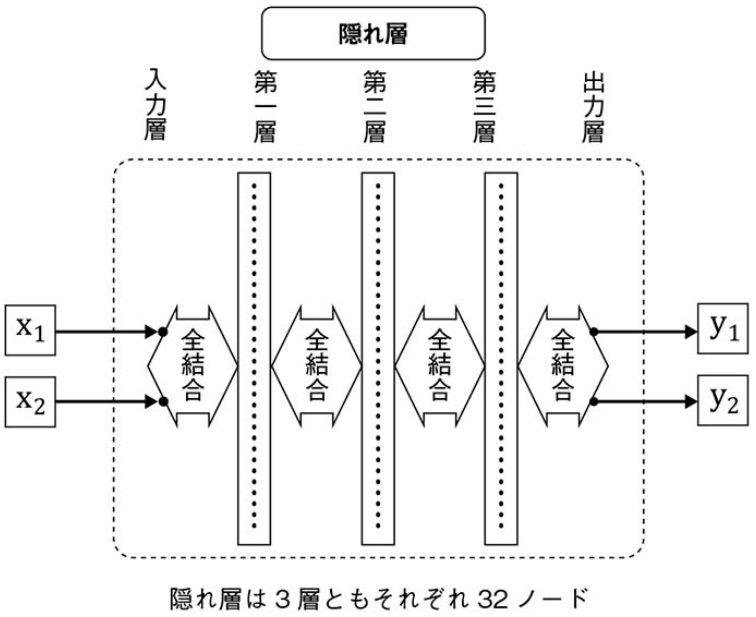


図3-20:学習で用いるニューラルネットワークの構成

上記のようなニューラルネットワークの構成でサンプルデータを学習させた場合、それぞれの隠れ層でどのような学習が行われているのかを見ていきます。それぞれの隠れ層の学習結果を可視化するために、各層のそれぞれのノードの**特徴量マップ**（**活性化マップ**とも呼ばれます）を用いることで、各層が学習した特徴を調べていきます。

まず、次の図3-21が隠れ層第1層の全ノードの特徴量マップです。これを見ると、どのノードも基本的に直線的な特徴をとらえていることがわかります。つまり、入力層に一番近いところでは、非常に単純な特徴をとらえていることが理解できます。

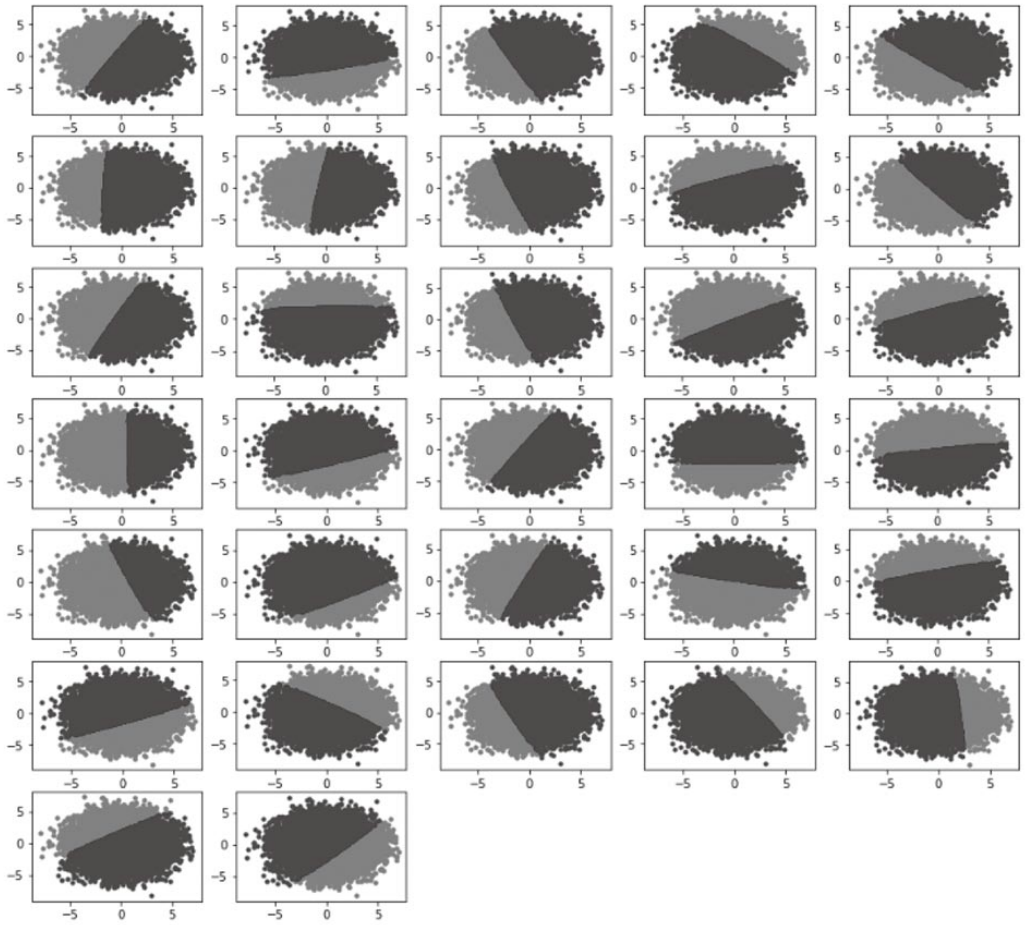


図3-21:隠れ層第1層の全ノードの特徴量マップ

次の図3-22が隠れ層第2層の特徴量マップです。こちらは第1層でとらえた特徴量が組み合わされているため、もう少し複雑な特徴をとらえていることがわかります。

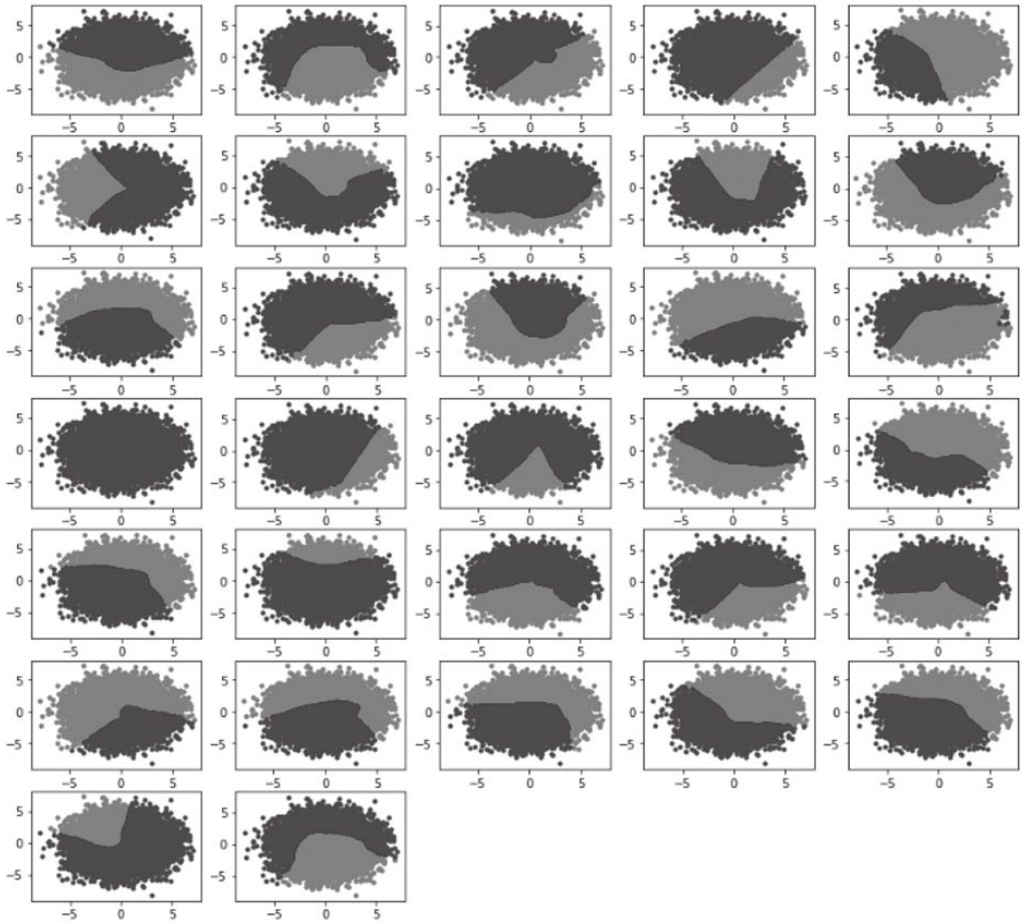


図3-22:隠れ層第2層の特徴量マップ

最後の図3-23が隠れ層第3層の特徴量マップになります。かなり正解値に近く、より複雑で全体的な特徴をとらえているのが理解できます。

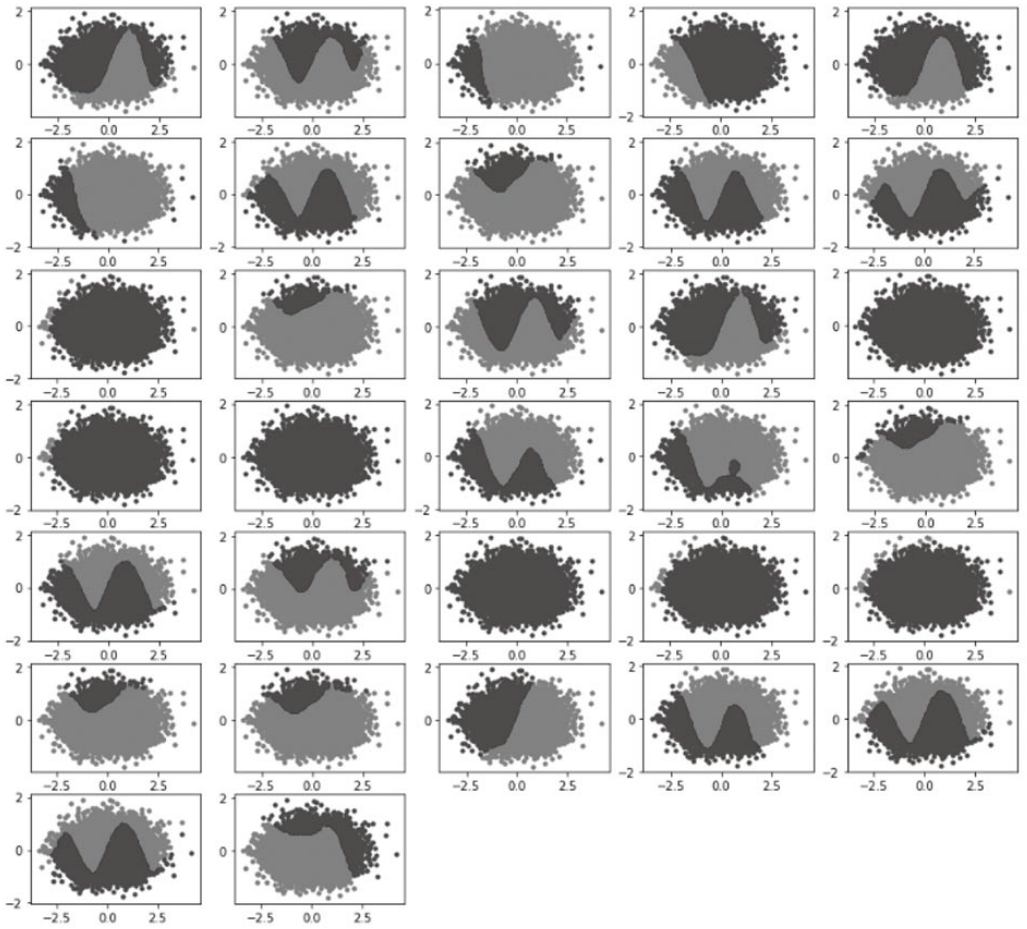


図3-23:隠れ層第3層の特徴量マップ

ここまで図に示した3つの隠れ層の特徴量マップの違いを整理すると、入力層に一番近い隠れ層である第1層では、ほぼ直線的で単純な特徴を学習していること、層が深くなるにつれて本来の出力の特徴の一部分に近い複雑な特徴を学習していることが理解できます。今回の例よりもさらに層が深い場合でも基本的な考え方は同じであり、層が深くなることで、さらに多くの複雑な特徴をとらえていけるようになることが容易にイメージできます。

3.7 ニューラルネットワークの学習の割合や回数に関する指定

勾配降下法のところですでに説明したとおり、ニューラルネットワークの学習では一般的に、ミニバッチ確率的勾配降下法を用います。そこで本節で説明する学習に必要なそれぞれのハイパーパラメータは、ミニバッチ確率的勾配降下法で使用するものと仮定して説明していきます。

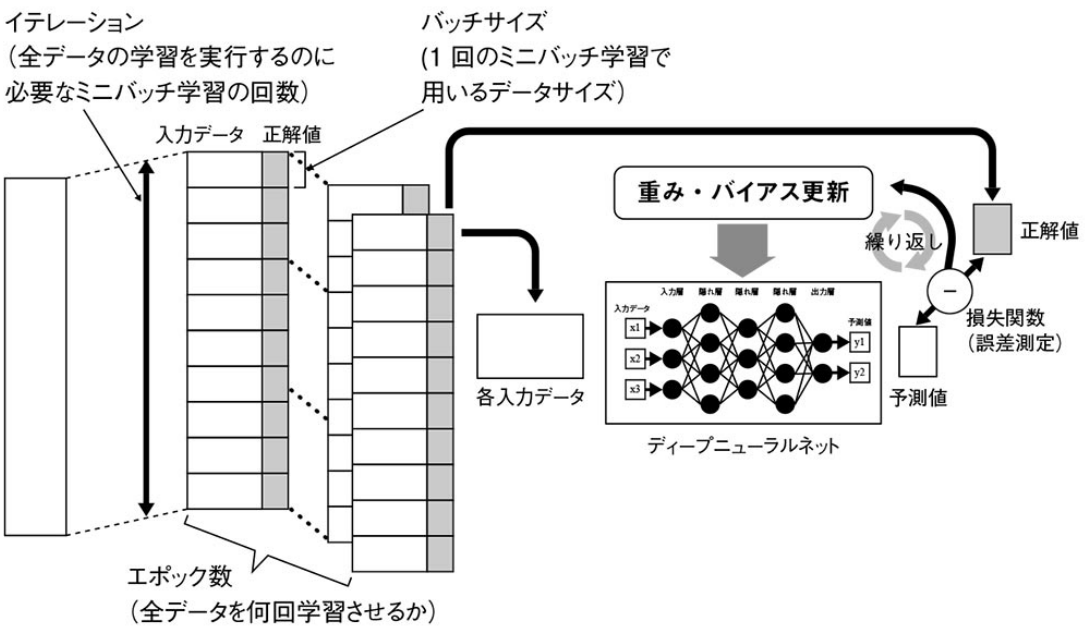


図3-24:学習で用いられるハイパーパラメータの概要

▶3.7.1 （ミニ）バッチサイズ

ミニバッチ確率的勾配降下法は、全訓練データを複数個に分割し、分割されたデータ単位で学習を1回実行し、パラメータを更新していきます。

このデータ量の分割単位は**バッチサイズ**もしくはミニバッチサイズと呼ばれますが、本書ではバッチサイズで統一します。

▶3.7.2 イテレーション数

ミニバッチ確率的勾配降下法で、全訓練データを1回学習するために必要な学習回数のことを**イテレーション数**と呼びます。

イテレーション数は、以下の式により、全訓練データ数とバッチサイズが決まれば自動的に求めることができます。

全訓練データ数＝バッチサイズ×イテレーション数（バッチ単位の分割数）

▶3.7.3 エポック数

学習では、全訓練データを何回も用います。そのため、全訓練データを用いる回数のことを**エポック数**と呼びます。全訓練データを一度使用したとしても、学習として十分ではないことがあるでしょう。ミニバッチ確率的勾配降下法では、エポックごとに、データはランダムにサンプリングされてミニバッチのデータに分割されるため、ミニバッチごとのデータの組み合わせが異なり、毎回違うデータの組み合わせで学習が行われます。これにより、全訓練データ自体は変わらなくても学習を繰り返すことで、精度がある程度上がっていきます。

ただし、エポック数を増やしすぎると大局的には同じ全訓練データを用いて訓練することになるので、過剰適合につながります。

▶3.7.4 学習率

1回の学習で更新させるパラメータの大きさ、つまり更新の幅を**学習率**と呼びます。山を下る例で考えると、下る道が決まった場合、次のチェックポイントをどれくらい先にするかを決める値と考えられます。10メートルごとにするか100メートルにするかでずいぶん結果が違いそうですよね。

この値を小さくすれば更新の幅が小さいので正確に学習が進んでいきますが、計算回数が多くなり学習の進み方が遅くなります。

更新の幅が大きいと学習のスピードは速いのですが、場合によっては正しく最

小値に収束しないことがあります。

ニューラルネットの学習に関連する基本的な説明は以上になります。

学習のフェーズの次は、評価のフェーズになります。評価のフェーズでは、ニューラルネットワークが未知のデータに対しても使えるだけの精度があるかをチェックします。評価で学ぶ内容は、先に示した図3-6のニューラルネットワークの処理プロセスの概要のうち、「学習済みモデルのテスト」および「モデルの実環境での評価」で用いられるものです。これで一連のディープラーニングの処理プロセスを進めるうえで必要な基本的な知識はすべてとなります。

3.8 ニューラルネットワークの評価

ニューラルネットワークの学習時に用いた訓練データでどんなに精度が出ても、本番環境の実際のデータで精度が出なければ意味がありません。先にも述べましたが、車の仮免と同じく、ニューラルネットワークでも、訓練データとは別に準備したテストデータに対しても精度が出ることを確認して初めて本番環境に実装可能となります。そのためには、免許皆伝となるべく何らかの評価を行う必要があります。

ニューラルネットワークの評価では、訓練用のデータとは別に準備されたテストデータを用いて精度を評価します。訓練データとテストデータのようにデータを入れ替えて評価する手法は**交差検証**と呼ばれ、ニューラルネットワークの予測値の精度の評価として一般的に用いられています。先の図3-6で説明したニューラルネットワークの処理プロセスの概要のうち、「学習済みモデルのテスト」と「モデルの実環境での評価」は、用いられるデータが違うだけで実行している内容はどちらも同じです。そのため、ここでは両者をニューラルネットワークの評価という一括りで説明していきます。

損失関数の節では、扱う問題が回帰か分類かの違いにより、使用する損失関

数が異なることを説明しました。損失関数が異なれば、その評価を行う評価関数も異なるのは明らかです。そこで、今から説明する評価関数についても回帰と分類に分けて行います。

▶3.8.1 回帰問題の評価関数

回帰問題で用いられる評価関数は基本的に損失関数と同じです。そのため、評価関数で用いられる数式も損失関数と同じものとなります。損失関数と評価関数で異なるのはその使われ方です。損失関数の場合は、計算結果を重みとバイアスの更新のパラメータとして利用するのに対し、評価関数はモデルの精度を調べるための評価結果として用いるのにとどまる点です。つまり、評価関数は、出来上がったモデルがほかのモデルと比べて精度が良いか悪いかの比較に用いますが、その結果を用いてモデルの精度向上につながる重みやバイアスに直接フィードバックされることはありません。

たとえば2つのモデルがあった場合、どちらのモデルの精度が高いのかを比べるのは簡単です。2つの評価関数の計算結果の数値の小さいほう、つまりより誤差が小さいものが良いモデルであると評価できます。

▶3.8.2 分類問題の評価関数

分類の場合は、回帰と違い、個々の予測値と正解値の間の直接的な誤差の大きさには特に意味がないため、別の方法を用いて比較を行う必要があります。基本的には、正解のクラスと予測された分類のクラスが合っているか否かの確認を行う必要があります。たとえば、学習フェーズで教師ありデータの評価を行う場合には、全訓練データに対して正解・不正解の数をカウントし一番正解が多いモデルを良いものとして評価するような仕組みが必要になります。

基本的な評価として**混合行列（Confusion Matrix）**を用いることになります。次の表は、二値の混合行列の例ですが、こちらを用いて混合行列の概要を説明していきます。

		予測	
		陽性 （Positive）	陰性 （Negative）
正解	陽性（Positive）	真陽性（True Positive）	偽陰性（False Negative）
	陰性（Negative）	偽陽性（False Positive）	真陰性（True Negative）

表3-2:二値の混合行列（Confusion Matrix）

上の表の行が正解のクラス、列が予測のクラスを表しています。正解および予測のいずれも陽性と陰性の二値が存在するため、それぞれサブクラスとして表記されています。表内では、真陽性と真陰性の部分が、正解値を正しく予測できたサンプルの数、偽陽性と偽陰性の部分が、誤って予測したサンプル数となるように表現されています。真と偽はサンプルの予測クラスが正解クラスと同じか否か、つまり的中したか否かを意味します。陽性と陰性は予測クラスのそれぞれで陽性と判断されたものなのか、陰性と判断されたものなのかを示しています。

それぞれは以下のような意味となります。

真陽性（TP）:実際に陽性であるものを陽性だと予測
偽陰性（FN）:実際に陽性であるものを陰性だと予測
偽陽性（FP）:実際に陰性であるものを陽性だと予測
真陰性（TN）:実際に陰性であるものを陰性だと予測

混合行列の各項目は、次の表とチャート（表3-3と図3-25）を合わせて見ると、単なる言葉の説明より理解しやすいので、ぜひしっかりと確認してみてください。

指標名	算出式	意味
正解率（Accuracy） 正確度とも呼ばれる	(真陽性＋真陰性) / (真陽性＋偽陽性＋真陰性＋偽陰性)	分類の精度を表す。全体の中で正しく分類できた率を示す。
再現率（Recall） 検出率とも呼ばれる	真陽性 / (真陽性＋偽陰性)	陽性のデータのうち陽性だと正しく分類された率を示す。
適合率（Precision） 精度とも呼ばれる	真陽性 / (真陽性＋偽陽性)	陽性と分類されたデータのうち真に陽性だったものの率を示す。
F1 スコア （F1 score）	F1 ＝ 2 * (適合率 * 再現率) / (適合率＋再現率)	再現率と適合率をバランス良く持ち合わせているかを示す指標。

表3-3:混合行列での評価指標

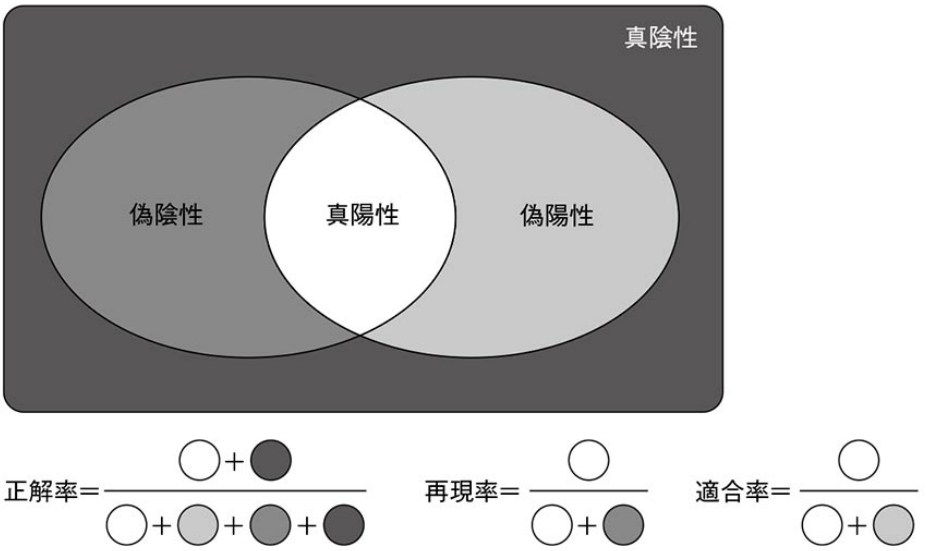


図3-25:各指標の関係性のチャート※1

一見すると、正解値の数だけカウントするだけではダメなのか、不思議に思われる読者の方がいらっしゃるかもしれません。たとえば医療現場では、ある病気に対して実際にかかっている患者だけを知りたい場合もあれば、確実にかかっていない患者だけを知りたい場合もあります。また、病気でない患者に陽性判定が出たり、逆に病気を有している患者に陰性判定が出たりする理由を知りたいときなど、さまざまな用途の違いに対して、正しく評価できるようにするために、このよう

※1　実践者向けディープラーニング勉強会 第1回 深層学習の基礎と導入 参照（https://www.slideshare.net/KazukiMotohashi2/ss-136324688）

な複数の指標が用いられます。

長い道のりでしたが、ニューラルネットワークに関する基本的な内容は以上です。

次の節では、本書の本題のディープラーニングで用いられるネットワークである、ディープニューラルネットについて詳しく見ていきます。ディープラーニングの大半の内容はニューラルネットワークの動きがベースとなっているため、今までの知識を用いながらディープラーニングの理解を深めていきましょう。

3.9 ディープニューラルネットとは

ディープラーニングで用いられるニューラルネットワークは、**ディープニューラルネット**と呼ばれます。すでにニューラルネットワークの学習を見てきましたが、ディープラーニングは、ニューラルネットワークより大きなネットワーク、つまり、層の数やノード数が多いネットワークを用いたニューラルネットワークの学習なので、基本的な動作に変わりはありません。その名のとおり、ニューラルネットワークを深くした（ディープにした）ものです。明確にどれだけ深くしたらディープニューラルネットとなるかという規定はありません。しかしながら少なくとも3層以上あり、それぞれの層が複数のノードで実装されているものと理解すれば大丈夫です。すでにいろいろなところで目にされていると思いますが、例としては次の図のようなものだとイメージしてください。

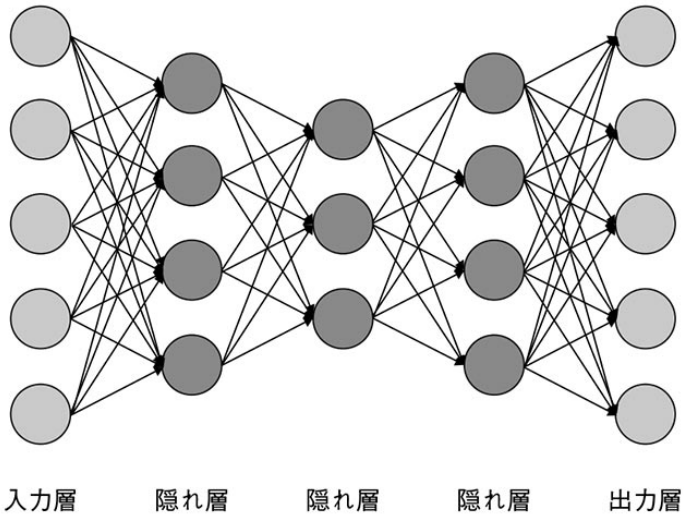


図3-26:ディープニューラルネットのイメージ

以降の説明では、実際にディープニューラルネットを構築していくための処理プロセスをイメージしながら、ディープニューラルネットならではの特徴や機能を見ていきます。これにより、ディープラーニングについての知識を深めていきます。ディープニューラルネットの具体的な説明に入る前に、ニューラルネットワークとディープニューラルネットの違いについてまとめておきます。なお、ディープニューラルネットはニューラルネットワークの一部であるため、ニューラルネットワークの特徴はディープニューラルネットにも共通する特徴になります。その点は注意してください。

◉――ニューラルネットワークの特徴

- **自動的にデータから知識やルールを獲得することが可能（機械学習全般の特徴）**
- **自動的に特徴量（適切な表現方法）を抽出することが可能**：入力データに含まれる、分析に必要な本質的な特徴量、内部表現、潜在表現などをマシンが自動的に抽出できる可能性があります。

◉――ディープニューラルネットならではの特徴

- **複雑な特徴も抽出・表現可能**：多量のノードを有するニューラルネットワークであり、さ

らには層が深くなることで結合方法も複雑になるため、複雑な入力データの特徴であっても細部の抽出もしくは表現が可能になります。

●**入力データの集約や圧縮が可能**：元データが持っている情報をできるだけ失うことなく（情報の損失なく）、うまくデータをほぐして表現することにより、より少ないデータで簡単な情報に集約したり圧縮したりすることもディープニューラルネットで実現可能になります。

▶3.9.1 ディープニューラルネットの処理プロセスと構成

本項では、ディープラーニングを実行するための処理プロセスと、そのプロセスの中でディープニューラルネットの実装に必要な構成要素について説明します。以下が処理プロセスの概要となります。

①データの準備・整形
- ・データの正規化
- ・データのテンソル化
- ・データ分割

②ディープニューラルネットのモデルの実装

②-1：モデルの形状に関する構成要素
- ・層の数
- ・各層のノード数
- ・ノード間の結合タイプ

②-2：各ノードに関する構成要素
- ・各ノードのタイプ
- ・活性化関数

③ディープニューラルネットの学習

③-1：学習パラメータ
- ・重み
- ・バイアス

③-2：学習目的を指定する構成要素

・損失関数

③-3：学習方法を指定する構成要素

・最適化

③-4：学習の割合や回数に関する指定要素

・学習率

・バッチサイズ

・エポック数

④ディープニューラルネットのモデルの評価

・評価関数

上記の処理プロセスの中で、学習の対象となる重みとバイアスは、**パラメータ**と呼ばれます。それに対して、学習する際に事前に手動などで設定する必要があるパラメータは**ハイパーパラメータ**と呼ばれ、パラメータとは区別されます。なお、重みとバイアスについても初期値は最初に手動などで設定する必要があるため、これらの初期値についてもハイパーパラメータに含まれることに注意してください。参考までに以下に主なパラメータとハイパーパラメータを表にしてまとめてみました。

ディープニューラルネットの構成要素の大分類	各要素
形状	層の数
	各層のノード数
	ノード間の結合タイプ
各ノード	ノードのタイプ
	活性化関数
学習	重みの初期値
	バイアスの初期値
	損失関数
	最適化
	評価関数
	学習率
	バッチサイズ
	エポック数

表3-4:ディープニューラルネットの基本的なハイパーパラメータリスト

ディープニューラルネットのパラメータ	要素
学習対象のパラメータ	重み
	バイアス

表3-5:ディープニューラルネットのパラメータリスト

これからディープラーニングでの各プロセスを、データ準備・整形から順番に説明していきます。

▶3.9.2 データの準備・整形

最初はデータの前処理になります。この処理の大部分はディープラーニングに限らずデータ分析全般で必要ですが、ここでは、ディープラーニングで特に重要な内容を中心に簡単に説明します。

▶3.9.2.1 データのベクトル化

ディープラーニングでデータを扱う際には、データをベクトル化するという作業

から開始します。**ベクトル化**とは、基本的に数値を個別の数値データとして扱うのではなく、何らかの関係性に基づきひとまとまりにして扱いやすくすることです。1次元のデータの場合にはベクトル、2次元の場合には行列、3次元以上の場合にはテンソルと呼ばれますが、複数次元でのデータ処理のためのデータ変換を総称してここではベクトル化と呼びます。ディープラーニングをはじめ機械学習では、データの相互の関連性を見ていく必要があります。そのため、ディープラーニングの実装にあたって最初に行う作業として、データをベクトル化するところから始めます。テキスト、画像、音声などの生データも、ベクトル化することで初めてディープラーニングで利用可能な形になります。

通常のデータベースなどのテーブルは行とカラムの2次元の行列になっているため、すんなりとイメージできる方が多いと思います。ここでは別の例として、ディープラーニングでよく用いられている画像データのベクトル化について説明します。

以下のデータは、機械学習の研究用データとしてよく知られているMNISTデータセットの中に存在するKuzushiji MNISTのサンプルデータの1つです。図の左の画像は、図の右のような28×28の2次元データとしてベクトル化して表現されています。図の例はグレースケール画像なので、各ピクセルの値としては0～255（0が黒、255が白）の範囲内の値が設定されています。

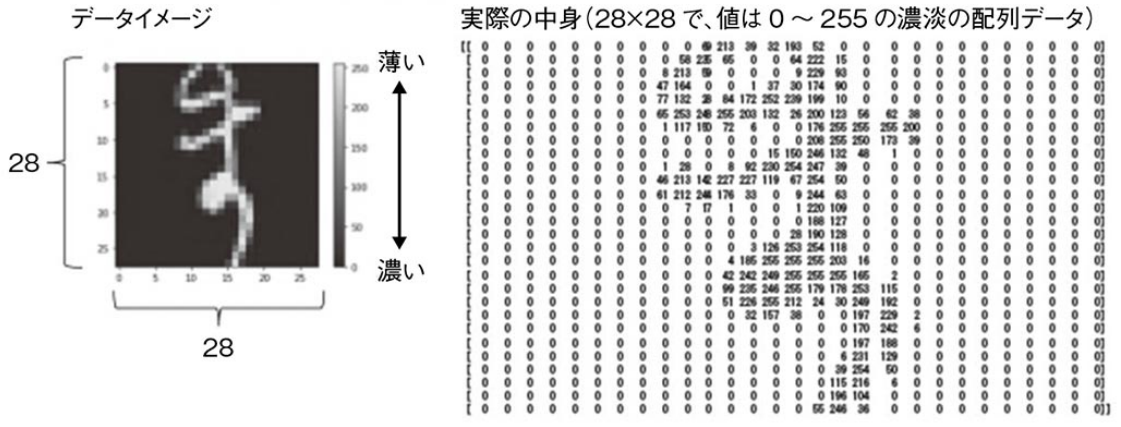


図3-27:Kuzushijiのサンプルデータ

▶3.9.2.2 ラベルエンコーディング

ラベルエンコーディングは、文字列のデータを数値変換する処理のことです。コンピュータのデータ処理では一般的に用いられており、ディープラーニングの前処理としても必要です。図3-28の左側がエンコード前のデータ、右がエンコード後のデータになっています。以下の例では、たとえば、jobというカラムはオリジナルのデータではhousemaid、services、adminなどという個別の職業の名前が入っていますが、右の表では、それぞれの職業を数値ラベルに置き換えることにより、数値演算で対処できるようにしています。

	age	job	marital	education	default	housing	loan
0	56	housemaid	married	basic.4y	no	no	no
1	57	services	married	high.school	unknown	no	no
2	37	services	married	high.school	no	yes	no
3	40	admin	married	basic.6y	no	no	no
4	56	services	married	high.school	no	no	yes

	age	job	marital	education	default	housing	loan
0	56	3	1	0	0	0	0
1	57	7	1	3	1	0	0
2	37	7	1	3	0	2	0
3	40	0	1	1	0	0	0
4	56	7	1	3	0	0	2

図3-28:ラベルエンコーディング

▶3.9.2.3 one-hot エンコーディング

one-hotエンコーディングは、特徴量がカテゴリデータである場合にone-hotのベクトルに変換する機能です。

ラベルエンコーディングのように、カテゴリ変数を数値にしただけでは、そこに本来の特徴量には存在しない数字の大小関係が生まれてしまい、コンピュータでは適切に扱えなくなってしまいます。たとえば、先ほどの例では、housemaidには3、servicesには7というラベルが付与されていますが、housemaidとservicesには数値的な3と7のような大小関係は存在していません。

そこで、カテゴリ変数を平等に扱えるようにするために、one-hotベクトル化が必要となります。図3-29の例は、Animalのうち、鳥、犬、猫というカテゴリを分類するために、任意の数字であるラベルエンコーディングとして0、1、2を割り振った場合とone-hotエンコーディングの違いになります。ラベルエンコーディングの場

合には、その数値の大小の関係性についても誤って学習してしまう可能性があります。つまり、鳥は犬より小さい、猫と比べるとさらに小さいなど、何の意味もない数字の比較関係が生じてしまい適切に処理できません。このような無意味な比較関係が生じないようにするために、one-hotエンコーディングを行います。つまり、図3-29の右の表のように、Animalの中のサブクラスであるbird、dog、catをそれぞれ別のカラム変数に変換するわけです。

回帰分析では、各カラムデータの数値の大小関係も学習してしまうため、ラベルエンコーディングは使用できません。回帰分析でカテゴリデータを用いたい場合には、one-hotエンコーディングが広く用いられています。

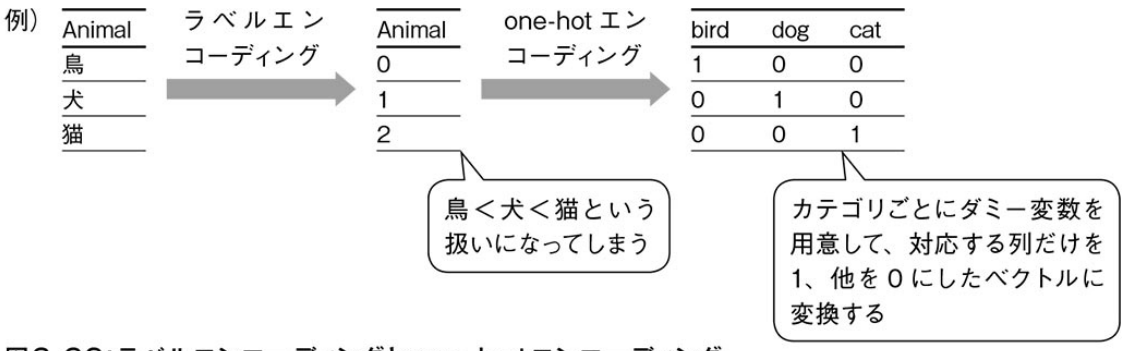


図3-29:ラベルエンコーディングとone-hotエンコーディング

▶3.9.2.4 データの標準化

データの**標準化**とは、平均が0、分散が1の正規分布（ガウス分布）となるように数値を変換することです。各特徴量のスケールが異なるデータで重みを更新しようとすると、スケールが小さいほうの更新幅が相対的に小さくなってしまいます。そのため、特徴量のスケールが小さいほうが本来の予測値を表現する特徴として意味があるデータであった場合、学習が遅くなったり先に進まなくなったりしてしまいます。このような状態を防ぐためにデータを標準化して、それぞれの特徴の間のスケールの差を小さくしておきます。同様の目的で、**正規化**（値を0～1に圧縮）することもあります。どちらを用いるかはケースバイケースですが、一般的に値のスケールが必ずしも明確ではない場合は、標準化を行います。

▶3.9.2.5　データ分割

ディープラーニングでは、図3-30のように、モデルの訓練用と学習済みモデルのテスト用に全データを分割します。場合によっては、モデルの実環境での検証ができないケースがありますが、その場合には3分割するようなケースも増えてきています。具体的には、図3-30のように一定の割合でランダムにデータを取り出し、分割します（通常、訓練データとテストデータの比率は8:2か7:3が多くなっています）。

以下は教師あり学習の場合の例になりますが、オリジナルのデータは、予測したい変数である目的変数の“正解値”と、目的変数を予測するための変数である“特徴量”（統計用語では説明変数）で構成されています。実際のディープラーニングの学習や評価では、ディープラーニングのモデルの学習用の入力データとして特徴量を用いて予測値を計算し、その計算結果である予測値と入力データの正解値を比較することで精度を求めていきます。そのため、モデルへの入力データとモデルの結果の比較データとして用いられる正解値は、実際のプログラムでは別変数として定義します。

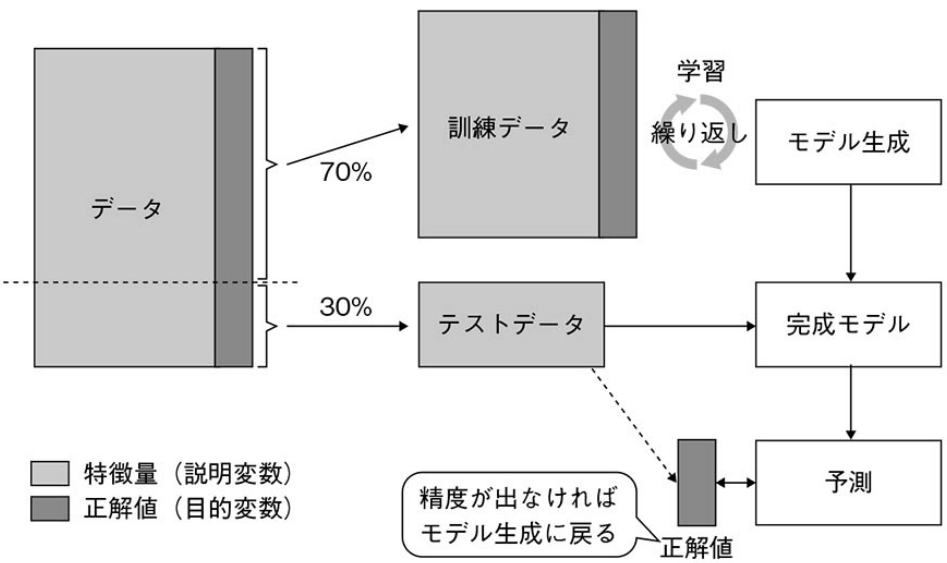


図3-30：訓練用とテスト用にデータを分割

データの準備にはそのほかにも、一般的なデータ分析でも必要となる外れ値やNull値を削除したり、予測に不要な特徴量を削除したりといった、さまざまなデータ準備をしなければならないケースがあります。実際のデータ準備自体で用いられる機能について深く知りたい方は、『機械学習のための特徴量エンジニアリング――その原理とPythonによる実践』(https://www.amazon.co.jp/dp/4873118689/)を参照されることをお勧めします。

▶3.9.3 ディープニューラルネットのモデルの実装

データの準備と整形が終わったら、ディープニューラルネットのモデルを実装していきます。ディープニューラルネットのモデルの形は、図3-31のように層の数、それぞれの層のノード数(ニューロン数)で指定します。

たとえば、図3-31は5層のニューラルネットワークであり、各層のノード数はそれぞれ3、4、3、4、2となっています。

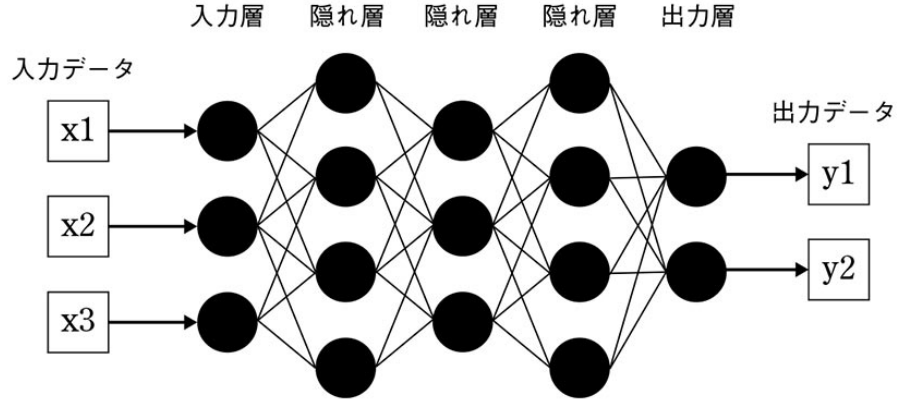


図3-31:ディープニューラルネットのモデルの形の例

さらに、ディープニューラルネットのそれぞれのノードでは、ノードの種類や活性化関数も定義する必要があります。

ノードの種類としては、分析の目的に従って適切なものを選択する必要があります。標準的な多層パーセプトロン、畳み込みニューラルネット、再帰型ニューラ

ルネットなど、さまざまなものを選択できます。本章では、最も基本的なニューラルネットワークの形である多層パーセプトロンの説明を行います。それ以外の主要なニューラルネットについては、第4章以降で紹介します。また活性化関数については、すでに3.4.3項で説明したとおり、解くべき問題に合わせて適切なものを選択します。

▶3.9.4 ディープニューラルネットにする意味

ここであらためて、ディープニューラルネットで層が深くなったりノード数が増えたりすることによるメリットについて考えてみます。

結論から言いますと、層を深くしたりノードを増やしたりすることで、複雑なルールを求めるような場合でも、実際に求めたいルールに一番近いものが利用可能になることです。これからその理由について、直感的な理解につながる内容を説明します。

学習とは、重みやバイアスを更新していき損失が最小になるものを見つけることでした。層やノードが増えるということは、重みやバイアスであるパラメータが増えるということです。言い換えると、関数で表現可能なパラメータが増えるということを意味します。さらに読み替えると、多数のモデルの候補の中から目的に最も適したものを1つ見つけるということになります。つまり、ディープニューラルネットで層やノードを増やすということは、表現可能となるモデルの選択肢が増えるということです。

たとえば、学習データを一番うまく分類できるモデルを100個の候補から選ぶ場合と1万個の候補から選ぶのを比較した場合、後者のほうがモデルの種類が多いために、データをより正しく分類できるものが見つかる可能性が高くなります。ただし、その一方で、見つかったモデルは偶然、学習データをうまく分類できているだけで、本当の意味でそのデータの特性をとらえていない可能性も高くなってしまう可能性を秘めています。

図3-32を見てください。左の○が層の数やノード数がより少ない場合に表現で

きるモデルの分布、右の○が層の数やノード数が左より多い場合に表現できるモデルの分布と考えてください。●が、求めるべき関数とします。左のように層やノードの数が不十分だと、実際に表現すべきモデルが含まれていません。対して、右の場合には、現状の層やノードの数で求めるべき関数が含まれているので、学習するためのモデルとしては十分だと言えます。

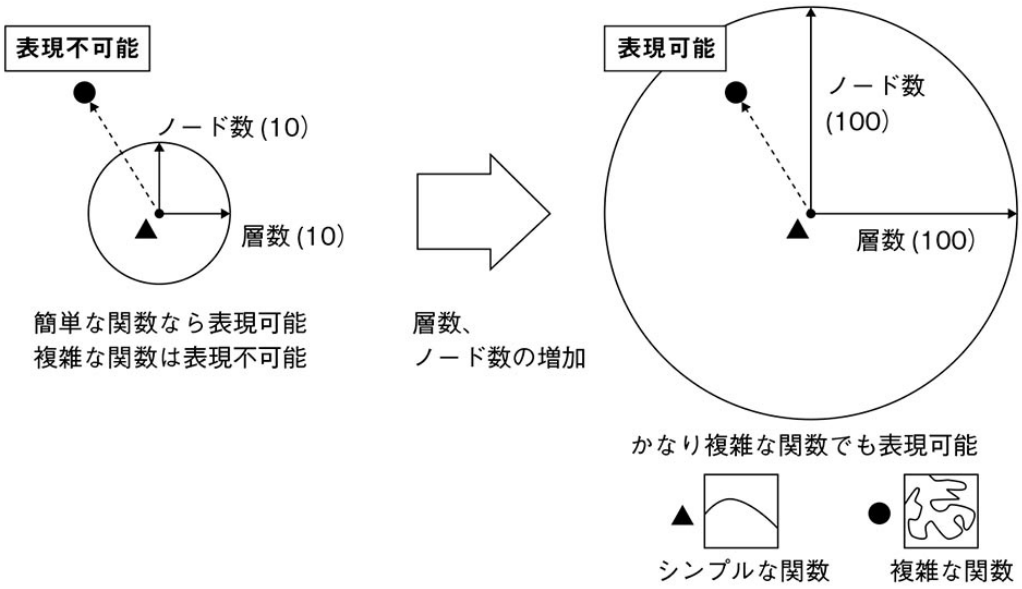


図3-32:層の数やノード数によって表現できるモデルの分布

これを見ると、層やノード数を増やしておけばディープラーニングがすべてうまくいくと思われるかもしれません。実際は、層やノード数を増やせば増やすほど指数関数的にコンピュータの計算量やメモリ使用量が増え、現実的な計算時間で終わらなくなってしまいます。さらには、関数の表現力が高くなりすぎて、データの細部の違いまで学習してしまうため、訓練データだと非常に高い精度で予測できますが、未知のデータに対しての予測結果が悪いというような、いわゆる過剰適合（過学習）になってしまう傾向があります。そのため、適度な層の数やノード数を設定することが重要です。一般的な分析においては、正解値を最もよく表現できる関数があらかじめわかっていないため、ノード数や層の数を最初に適切に設定

する法則は特にありません。そのため、開発現場では、ある程度過去の経験則に則り、ノード数や層の数を決め、学習を進めていきながらその数を調整していくという方策をとっていくことになります。

モデルのハイパーパラメータを設定して学習を始める際、最初は層の数やノードの数を適当に多めにとるのが一般的です。その場合、学習を開始した直後では間違いなく最適な関数（近似）よりも複雑で細かすぎる関数となるため、過剰適合の状態になります。評価関数を用いて、得られた学習結果を評価しながら、十分な精度に到達するまで層の数やノード数を調整していくことで、課題に適したモデルをチューニングしていきます。

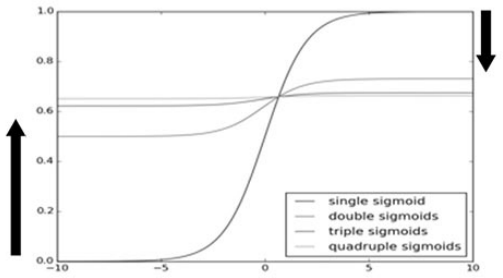
今までのところで、ディープニューラルネットにする意味と、ハイパーパラメータの設定の方針に関する情報を説明しました。次の項で学習の動作について詳しく説明しますが、その前に、ディープニューラルネットの場合に顕在化する可能性がある勾配消失という問題について説明します。

■ COLUMN　勾配消失問題とは

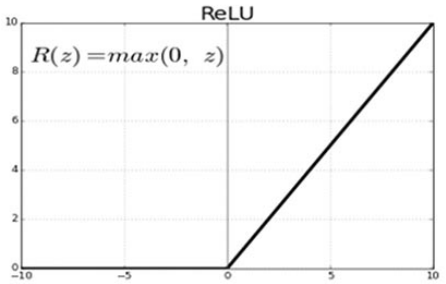
図3-33左からわかるように、シグモイド関数やtanh関数などの活性化関数を複数の層にわたって用いた場合、繰り返し用いられることで、次第にどの入力データに対しても出力値が0に収束してしまうという現象が起きます。ディープニューラルネットの学習は、関数の勾配をもとにパラメータの更新を行うため、図3-33左のように関数がすべて0付近に収束してしまうと、関数の勾配がなくなってしまい、学習が進まないという状態に陥ります。このように層が深くなることで、学習が進まなくなってしまう状態は、**勾配消失**と呼ばれます。これとは逆に、損失が指数関数的に大きくなり発散してしまう状態は、**勾配爆発**と呼ばれます。

シグモイド関数を隠れ層の活性化関数として用いた場合には、勾配消失の問題が発生してしまうため、近年は用いられなくなりました。それに対してReLU関数は複数回用いても、図3-33右のように関数の形は不変で、勾配消失の問題が起きることがないため、近年は隠れ層の活性化関数として一般的に用いられています。

f(x)：元の関数（シグモイド、tanh）を複数回用いると、
勾配消失問題が発生する



ReLU の場合は複数回用いても、
勾配消失問題は発生しない



注）あくまで直感的なイメージをつかめるようにするための説明
左のグラフはシグモイド関数のイメージ

図3-33：活性化関数と勾配消失問題の関係

3.10 ディープニューラルネットの学習

ディープラーニングの学習手法で用いられる**誤差逆伝播法**について説明します。今日、ディープラーニングがこれほどまでに浸透した1つの重要な要因は、誤差逆伝播法の存在です。誤差逆伝播法はパラメータを逐次追っていくのに時間がかかり一見非常に複雑ですが、ディープニューラルネットの学習を高速に実行するためのコアのアルゴリズムです。非常に重要な概念なので、そのメカニズムについて押さえておきましょう。ディープラーニングのさまざまな書籍で、誤差逆伝播法に関する数式の細かい解釈がなされています。誤差逆伝播法は、詳しく議論していくとなると内容が豊富でかつ細かい数式を理解していくのに時間と忍耐を要します。そのため、誤差逆伝播法の正確な理解のところでつまずいてディープニューラルネットの利用をやめてしまった人を何人か見てきています。本書では、実践するのに最低限の知識と押さえたい内容としてのバランスを考え、誤差逆伝播法の数式的な説明はできるだけ割愛します。ただし実践をこなしていくうちに、誤差逆伝播法の数学的背景や動作の詳細が必要になる場面が出てきます。今後、そのような必要が生じた際には、誤差逆伝播法についてさらに詳しく説明さ

れた専門書を参照されることをお勧めします。

なお、私の個人的な経験ですが、既存のよく知られたディープニューラルネットのアルゴリズムを扱う際に、具体的な数式を毎回頭に浮かべながら実践することはありません（アルゴリズムの選択が極めてうまくできていない場合は別ですが）。論文で新しいものを理解するとき、もしくは自分で一からアルゴリズムを実装するときに考える程度なので、初めてディープラーニングに触れる方は大雑把な誤差逆伝播法の動作イメージを持つことが重要だと考えています。

前置きがすっかり長くなってしまいましたが、これから誤差逆伝播法を見ていきましょう。

▶3.10.1 誤差逆伝播法とは

ディープラーニングもニューラルネットワークと根本的な仕組みは同じなので、パラメータを調整して損失関数の値が最小になるように最適化を行っていくことになります。また、損失の値を最小にするためには、勾配降下法を用いて、パラメータを変化させるべき方向や分量を調べていくことはすでに説明しました。勾配降下法は、実際の計算では、誤差の接線の傾きの方向と大きさを調べ、さらに学習率という1回あたりの学習量を掛け合わせることで、パラメータの値を更新していくのでした。最終的に最適解にいたる（誤差が最小になる）までこの操作を繰り返す点についても、ディープラーニングとニューラルネットワークにおいて変わりはありません。

ディープラーニングがニューラルネットと異なる部分は、層が深くなることです。層が深いとどうなるでしょうか。入力から出力までの層が増えるため、各重みやバイアスの調整が複雑になります。ニューラルネットワークのように層が少ないネットワークであれば、力技で最適な重みとバイアスを見つけることが可能かもしれません。しかし、近年のディープニューラルネットのように、層が100層を超えたり各ノード数も100を超えたりするものは、調整すべきパラメータ数が1万個を超えたりするため、手動で最適値を求めることは絶対に不可能です。現実的な計算時

間内で高速かつ半自動的に、多量の重みとバイアスを計算する仕組みが必要です。その課題を解決するために用いられるようになったのが、今から説明する誤差逆伝播法です。

図3-34のような、手書きの文字の「あ」から「お」のひらがなを予測するディープニューラルネットを考えます。実際のパラメータの更新では学習率についても考慮しなくてはなりませんが、ここでは、誤差逆伝播法の基本的な動作の説明に際しては重大な影響がないので、学習率については説明しません。もし気持ちが悪ければ、学習率を1に設定したものと仮定して理解してください。あくまでもこのディープニューラルネットは誤差逆伝播法をわかりやすく説明するために非常にシンプルにしたもので、具体的な例ではないことに留意してください。

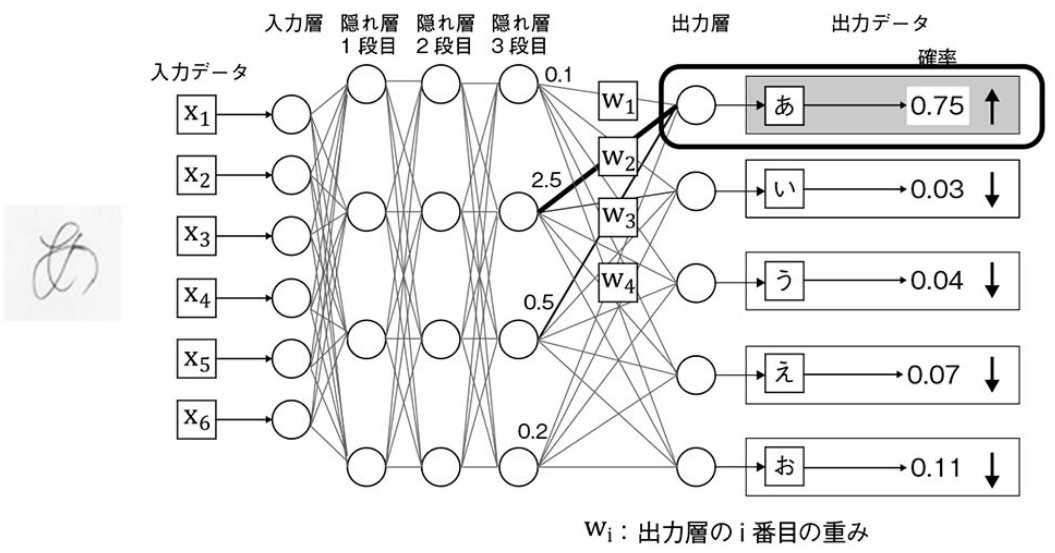


図3-34:出力層に注目した誤差逆伝播法の説明

図3-34の例では、ディープニューラルネットのモデルに対して、左端の「あ」の手書き文字のサンプルデータが入力されています。この出力結果としては、「あ」である確率が0.75（75%）であると判断しています。今回の場合、「あ」が正解値なので、予測値としては出力層の一番上のノードの確率をもっと上げ、他の出力層の確率を下げることで、精度が上がるようにパラメータを更新したいわけです。

具体的な動きで理解してもらうために、図の「あ」の出力層の前段である隠れ層3段目のノードの出力が一番上から0.1、2.5、0.5、0.2であったとします。その場合、3段目のノードの中で出力層の変化に一番大きな影響を与えるのは、上から2番目のノードであるとわかります。なぜなら同じ段の他のノードより出力値が大きく、たとえば一番上のノードと比較した場合、一番上のノードの出力の値は0.1であるのに対し、2番目のノードは2.5であるため、単純に25倍の影響が与えられると考えられるからです。このことが意味しているのは、「あ」という特徴を一番とらえている3段目の隠れ層のノードは上から2番目だということです。予測値を正解値に近づけるためには、このように正解値に近い特徴を大きく示しているノードの重みの数値を上げ、外れているものの重みの数値を抑えていくことでさらに正解値に近いものにしていきます。

効果的に正解値に近づけるには、前段のノードのうち一番影響力があるノードの重みを変更することが先決です。つまり、上から2番目のノードの重みであるw_2の変化量が一番大きくなるような方向で更新し、ほかのノードの重みについても、同様のルールで効果的に重みを調整する方法を見つけたいのです。この話のくだりは何かに似ていませんか。勘のいい読者の方ならお気づきのことと思いますが、勾配降下法です。勾配降下法では、いろいろな下り道の中で一番勾配がきついものを選ぶことでした。誤差逆伝播法では、前段のノードの重みの更新において、一番効率良く正しい方向に出力の変化を与えるものを選択するために勾配降下法を利用します。

図の「あ」の手書き文字を入力データとした場合、出力層の一番上のノードの予測値として文字「あ」である確率を効率良く上げるために1つのノードの重みを調整する動作についての説明は以上になります。

図の場合、出力層の一番上のノード以外の出力層の値は逆に低い値になるように調整しなければなりません。そのためにやることは、今説明したことと同じです。違いは出力層の値を上げるか下げるかであるため、勾配降下法を用いて他の出力層のノードの値が低くなるように重みを調整していきます。

次に、その前段にある隠れ層3段目のパラメータの更新の方法について説明します。

実は非常にシンプルです。今までの説明では最後の出力層に注目し、その1段前にあたる隠れ層3段目の出力値からのパラメータの調整を検討しました。今回は、図3-35のように、そのさらに前段にあたる隠れ層2段目のノードに注目してみます。そうするとこの場合、隠れ層2段目の出力のうち、一番下のノードの1.8が一番よく「あ」の特徴を示していると想定され、次に下から2番目のノードである0.8が続くと思われます。そのような特徴を伝搬するために、こちらでも勾配降下法を用いて重みを更新すればよいのです。

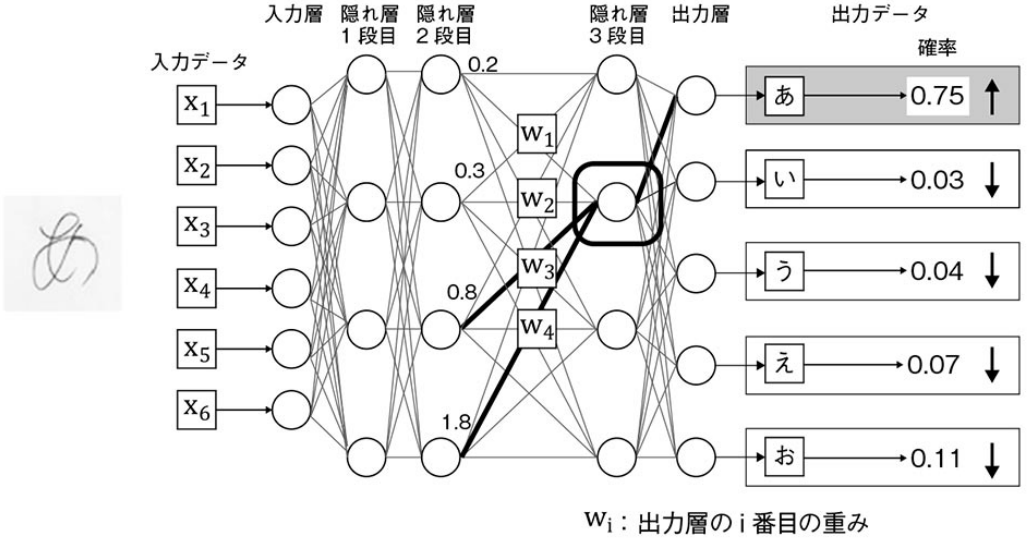


図3-35：隠れ層3段目に注目した誤差逆伝播法の説明

つまり、実行していることは、出力層がその前段に変わっただけであり、同じ勾配降下法を用いて出力層で一番正しい予測値に近づけるために、重みの調整を行っています。さらに、その前段の隠れ層1段目に対しても、基本的に同様の処理である勾配降下法による計算を繰り返していくことにより、損失関数から得られた損失の値の誤差を最小値にするために重みを更新していきます。

このとき1点だけ注意が必要です。出力層のパラメータの更新は、まさに出力層の直前の隠れ層の出力で推し量ることができました。しかし、もう1つ前段の隠れ

層3段目のパラメータの更新を考えた場合、最終的な出力層からどの程度の影響を受けていることになるのでしょうか?

図3-35のように、隠れ層2段目の上から4番目のノードの出力である1.8が出力層の値にどの程度の影響を与えるかを考える場合、出力層との間には隠れ層3段目のパラメータと活性化関数が挟まっている関係で、出力層からの影響が間接的なものにならざるをえません。つまり、隠れ層2段目の出力の影響は、隠れ層3段目の影響と出力層の影響の両方を考慮することになります。この影響度は、連鎖律と呼ばれるそれぞれの影響度の掛け算で表すことができます。連鎖律の概要については、後述する「3.10.2.1 計算グラフを用いた偏微分・連鎖律の理解」を参照してください。このように誤差逆伝播法では、出力のほうからこの連鎖律を当てはめることで、各ノードの出力層に対して損失関数に与える誤差の影響度が調整可能となります。

さらに前段にあたる隠れ層1段目のパラメータを調整する場合には、出力層、隠れ層3段目、隠れ層2段目それぞれの影響度を掛け合わせることによって表現可能となります。これを一般化して考えると、さらに深い隠れ層を持つディープニューラルネットについても、各ノードの影響度にあたる勾配は勾配降下法に基づき計算し、各ノードのパラメータはそれぞれの勾配の結果に連鎖律の法則を当てはめ、出力層から各ノードまで掛け合わせることで計算が可能になります。この仕組みにより、最終的には出力層のノードから入力層のノードにいたるまでのすべてのパラメータの更新が可能となります。この手法が**誤差逆伝播法**(Back Propagation)と呼ばれます。誤差逆伝播法が逆と名付けられている理由は、入力データから出力である予測値を求めるのを順方向と考えた場合、今回のパラメータの調整では、出力側から入力側という逆方向に計算をさかのぼるためです。

囲碁でもそうですが、いきなり100手先まで読むのは素人では不可能です。一手一手確実に読み解くルールが確立していれば、それを利用することで20手先であろうが1000手先であろうが何手先でも読み解くことが可能になります。残念ながら囲碁ではこのようなルールは存在しませんが、損失関数の最適化では、誤

差逆伝播法のおかげで非常に深い層のニューラルネットワークでも、すべてのパラメータの更新ができるようになりました。ディープラーニングで誤差逆伝播法が一般的に利用可能になる前は、このような深い層でのパラメータの更新は神業に近く、初学者が簡単にチューニングできるものではありませんでした。つまり、誤差逆伝播法の発明は、ディープラーニングが今日の広がりを見せるのに大きな貢献を果たした要因の1つなのです。

ここまでで、1つのサンプルデータを用いた誤差逆伝播法によるパラメータ更新の動作について説明しました。特に数式を使わなくても誤差逆伝播法をイメージできたのではないでしょうか。

最後に多くのサンプルデータを用いた場合の誤差逆伝播法のイメージを簡単に説明します。

こちらもいたってシンプルです。この後の図に示すように、各サンプルデータで計算されたパラメータをノードごとに合計した後、サンプル数で割ることにより平均値を算出し、その値を使って各ノードのパラメータを更新するだけです。

▶3.10.2 数式を用いた誤差逆伝播法についての補足

さらに正確な動作をとらえるうえでは数学の力を借りるのが便利です。本項の説明では、偏微分とそれに関わる数学的な法則を用います。これらの説明には理系の大学1年生で履修する内容が含まれています。法則自体の細かい説明を行うことは本書の趣旨と異なるため、法則の正確なロジックではなく法則を活用するという視点に立って説明していきます。この法則自身はかなりシンプルですが、どうしても数学にアレルギーがある方は本項を読み飛ばしても次章以降の理解に支障がないように配慮していますので、その点はご安心ください。

ここでは、細かい数式の正確な説明ではなく、誤差逆伝播法について理解するための補助的な役割を果たす動作イメージについて紹介します。

▶3.10.2.1 計算グラフを用いた偏微分・連鎖律の理解

最初に、誤差逆伝播法を説明するのに必要となる偏微分と連鎖律がどのように用いられるかについて、1つの例をもとに見ていきます。

計算グラフの紹介で使った(a+b)×(b−c)の数式が、この後の図3-36に示すようにそれぞれの入力値が変わることで、計算結果にどの程度の影響を与えるのかを知りたいとします。理解しやすいように、具体的な数値で説明していきます。

微分とは、簡単に言えば関数の変化の割合です。

たとえば今回使用する計算式で、aの値が変化したときにfの値がどのくらいの割合で変化するのか、つまりどの程度影響を受けるのかを調べるような場合に微分を用います。

今回の例のように、複数のパラメータが存在する場合には、a以外の入力値は固定する、つまりa以外は定数とみなし、aの値が6から7に変化した場合にfの値がどれくらい変化するのかを調べます。ちなみに、このようにいくつものパラメータがある場合、1つのパラメータに注目してその他のパラメータは定数とみなして微分の計算を行うことを**偏微分**と呼びます。さらに、図3-36のようにaの出力結果とfとの間に、もう1つdという計算のステージが挟まる場合には、最初にaとdの変化の割合を調べ、さらにdとfの変化の割合を調べて、それぞれの変化の割合を掛け算することで、aからfへの変化の割合が求められます。このようにパラメータの入力から出力までに計算が複数のステージで構成される場合には、各ステージの偏微分を計算して、すべてのステージの偏微分を掛け合わせることで、あるパラメータの入力の変化によって出力に与える変化の割合が計算可能となります。1つのパラメータが通るルートの各ステージの偏微分を掛け算することで、全ルートの変化の割合を表すことができる、という規則が**連鎖律**と呼ばれます。

最初は、微分の式を用いずに図の数値を具体的に計算していくことで感覚的に理解できるように説明していきます。

たとえば、aを6から7に変化させた場合には、dの値は10から11に増加し、fの値は20から22に増加します。つまりaが1増加するとfは2増加します。このように

して1つ1つのパラメータの変化を追っていけばすべての変化の割合を調べることができます。ただこのように1つ1つのパラメータの変化を追っていくのは非常に効率が悪いため、より効率的に一般式で解けるようにする方法が、先に説明した偏微分と連鎖律になります。

ちなみに、変化の割合は、関数の勾配（傾き）を意味するため、これからは**勾配（傾き）**と呼びます。

実際にaからdの勾配を示す偏微分の式は、図の $\frac{\partial d}{\partial a}=1$ で記載されているとおり1になります。また、dからfの勾配は同様に $\frac{\partial f}{\partial d}=2$ で2となるため、aからfの傾きは連鎖律を当てはめることで1×2=2となり、先ほどの実際の計算と合っていることが確認できます。その他のそれぞれのルートのプロセスについても偏微分と連鎖律を利用することで、求めたいルートの変化の割合である勾配を知ることができます。

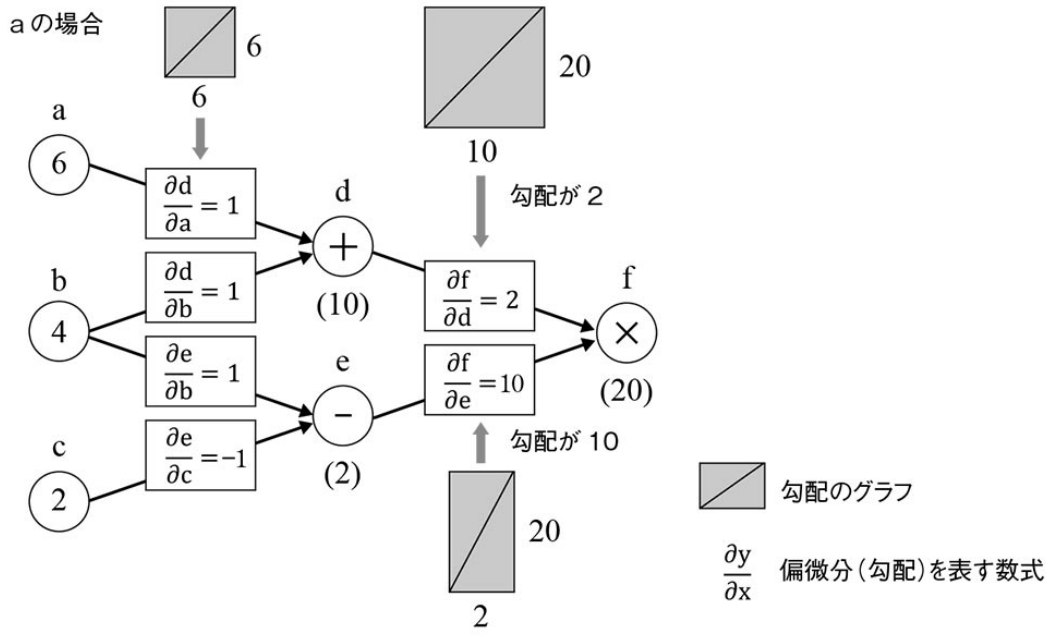


図3-36:偏微分の例

上記の例では、入力側から出力に与える影響を調べました。以上で、具体的な偏微分と連鎖律がイメージできたと思います。

▶3.10.3 誤差逆伝播法の数式的理解

偏微分と連鎖律がイメージできたところで、誤差逆伝播法を数式的に理解していきましょう。

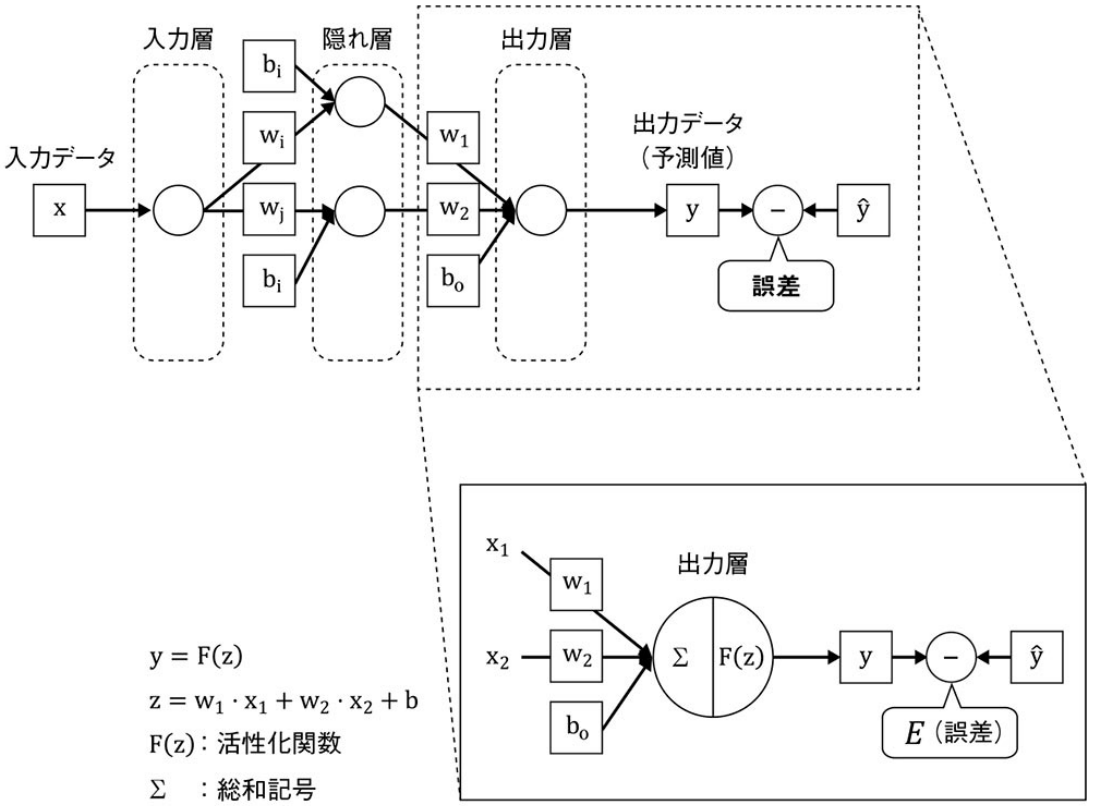


図3-37：出力層の前後の構成

誤差逆伝播法の詳細な動きを説明するために、まず図3-37の右にある出力層のノードに注目し、出力層の前段の重みの更新について考えます。なお、重みの更新は以下の数式で定義されます。

$$w_{t+1} = w_t - \gamma \frac{\partial E}{\partial w_t}$$

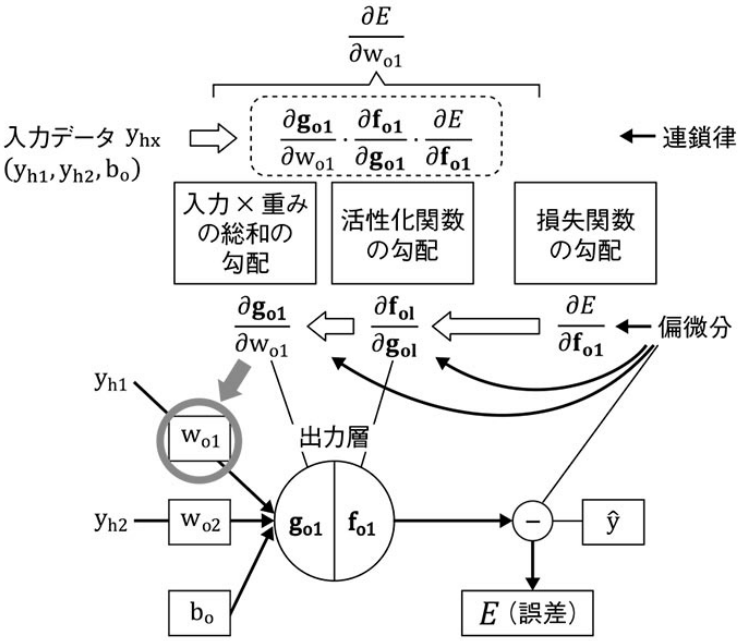
数式3-5：重み更新の公式

tは学習の更新タイミング、wは重み、γは学習率、Eは損失関数です。

それでは、誤差逆伝播法で各重みを更新するためにどのようにそれぞれの勾配を求めるか見ていくことにしましょう。

偏微分と連鎖律を利用できるので、難しくはありません。図3-38のように、グレーの太丸で示されるw_{o1}の重みに関して勾配を計算する場合、損失Eから重みw_{o1}までさかのぼっていくのに、間に出力層の活性化関数と総和が挟まっています。そのため、それぞれの偏微分をとり、図3-38のようにそれぞれの関数の勾配を求め、連鎖律をもとにそれぞれの勾配を掛け合わせることで、損失Eが重みw_{o1}に与える影響の割合となる偏微分$\frac{\partial E}{\partial w_{o1}}$が求まります。

具体的には、図3-38の点線枠で示されているとおり、損失関数の勾配である$\frac{\partial E}{\partial \mathbf{f_{o1}}}$、出力層の活性化関数の勾配である$\frac{\partial \mathbf{f_{ol}}}{\partial \mathbf{g_{ol}}}$、出力層の各入力（前段の隠れ層の出力）に重みを掛けたものの総和に対する重みの勾配である$\frac{\partial \mathbf{g_{o1}}}{\partial w_{o1}}$を偏微分として求め、連鎖律を用いて各偏微分の結果を掛け合わせることで、重みw_{o1}に対する損失の勾配である$\frac{\partial E}{\partial w_{o1}}$が求まります。



誤差関数：E
正解値：$\hat{y}$
出力層の入力（前段の隠れ層の出力）：y_{hx}
出力層の総和：$g_{o1}(w_{ol} * y_{h1} + w_{o2} * y_{h2} + b_o)$
出力層の重み：w_{ox}
出力層のバイアス：b_o
出力層の活性化関数：f_{o1}

図3-38:出力層の重みw_{o1}に対する損失の勾配

以上が図の出力層の前の重みw_{o1}の更新のメカニズムです。

次に、図3-39の隠れ層hの重みw_{h1}の更新のメカニズムを調べます。隠れ層における計算も基本的には、出力層の重み計算で用いた計算方法がそのまま使えます。出力層でも隠れ層でも、図のように損失関数の勾配、活性化関数の勾配、入力×重みの総和の勾配を計算するという処理のフレームワークは同じであることがわかります。

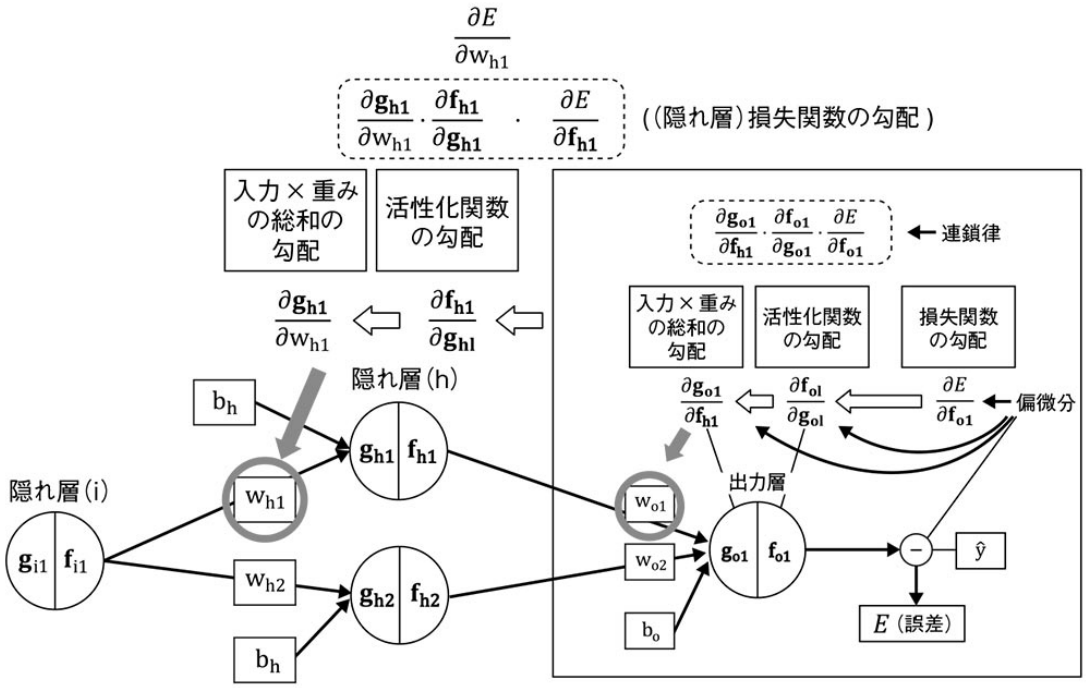


図3-39:隠れ層hの重みw_{h1}の勾配

実際に、出力層と隠れ層hの重みの勾配の計算式を図3-40のように切り出して見比べてみると、式の原型は一緒で、出力層では引数としてo1が使用されているのに対し、隠れ層hでは、h1が用いられている点が違うだけであることがわかります。

$$\frac{\partial E}{\partial w_{h1}}$$

$$\frac{\partial g_{h1}}{\partial w_{h1}} \cdot \frac{\partial f_{h1}}{\partial g_{h1}} \cdot \frac{\partial E}{\partial f_{hl}}$$

隠れ層hの重みw_{h1}の勾配

$$\frac{\partial E}{\partial w_{o1}}$$

$$\frac{\partial g_{o1}}{\partial w_{o1}} \cdot \frac{\partial f_{o1}}{\partial g_{o1}} \cdot \frac{\partial E}{\partial f_{o1}}$$

出力層の重みw_{o1}の勾配

図3-40:出力層と隠れ層hの計算式

また、隠れ層hでは、上記の$\frac{\partial E}{\partial w_{h1}}$のうち、$\frac{\partial E}{\partial f_{h1}}$（図3-41ではグレーの太丸部分）については、図3-41の右の式のように展開でき、これにも連鎖律が当てはまるため、$\frac{\partial E}{\partial w_{h1}}$は最終的に図3-41の最終式のように展開することができます。

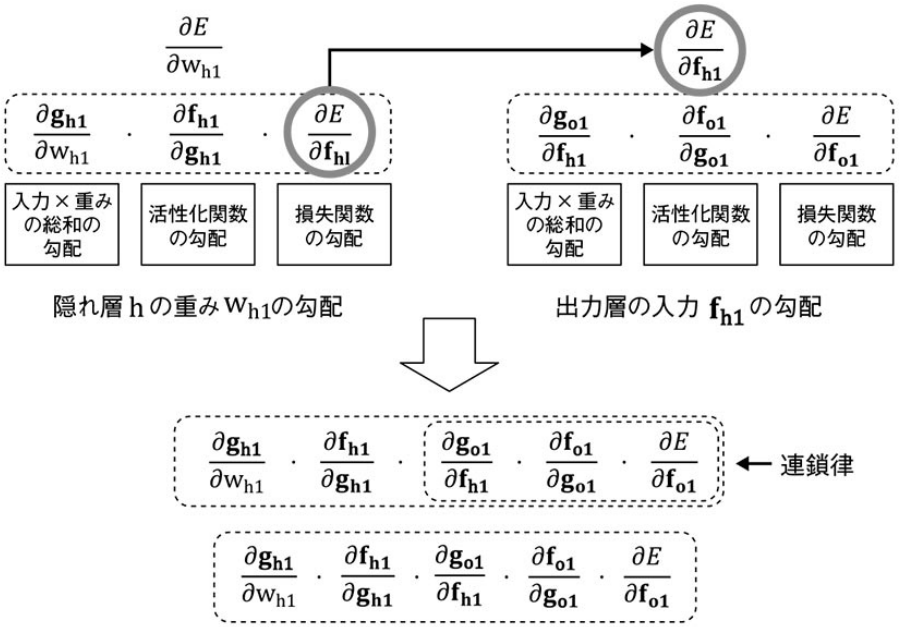


図3-41：隠れ層hの重みw_{h1}の勾配の展開による更新

もしさらに隠れ層が複数段存在する場合にもまったく同じ考え方を用いることが可能です。

次の図3-42で、図3-41のニューラルネットワークにさらに一段前に隠れ層iが実装された場合を考えます。これらの図の違いを見比べると、図3-42では、図3-41の隠れ層hであったものがiに代わり、出力層oが隠れ層hに代わっただけで、処理のフレームワークはまったく一緒なのがおわかりいただけるでしょう。このような処理を繰り返していくことで、さらに深い層の隠れ層であっても誤差逆伝播法を用いれば、それぞれの層の重みの勾配を求めることが可能となります。

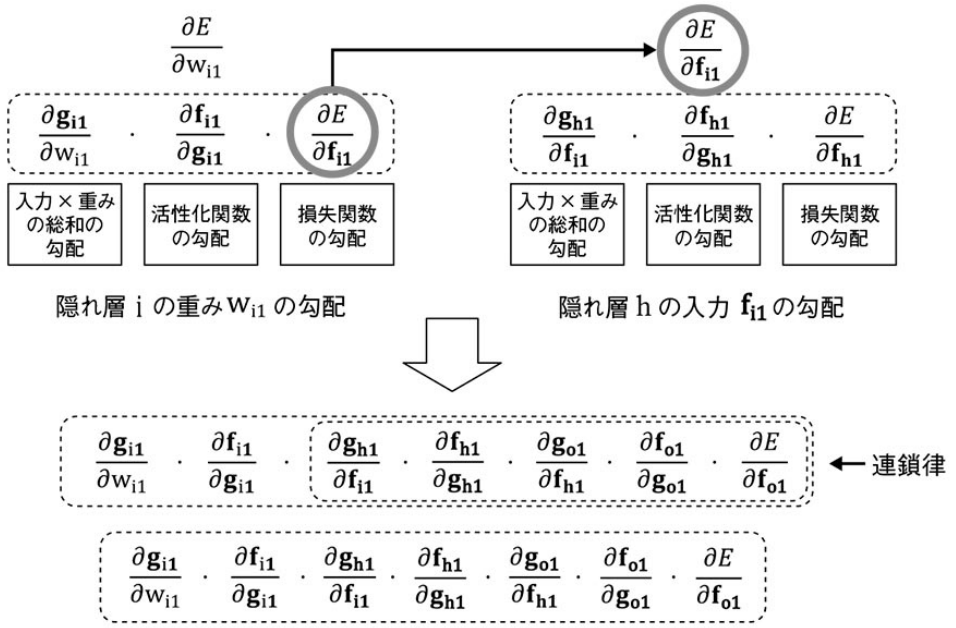


図3-42:隠れ層iの重みw_{i1}の勾配

今までの例では、説明を簡単にするために、1つのニューラルネットワークの経路に対しての勾配計算を追っていきました。しかし実際のニューラルネットワークでそれぞれの重みの更新量を求める際には、下記にあるようにさまざまなニューラルネットワークの経路すべてに対して同様の勾配計算をしていく必要があります。

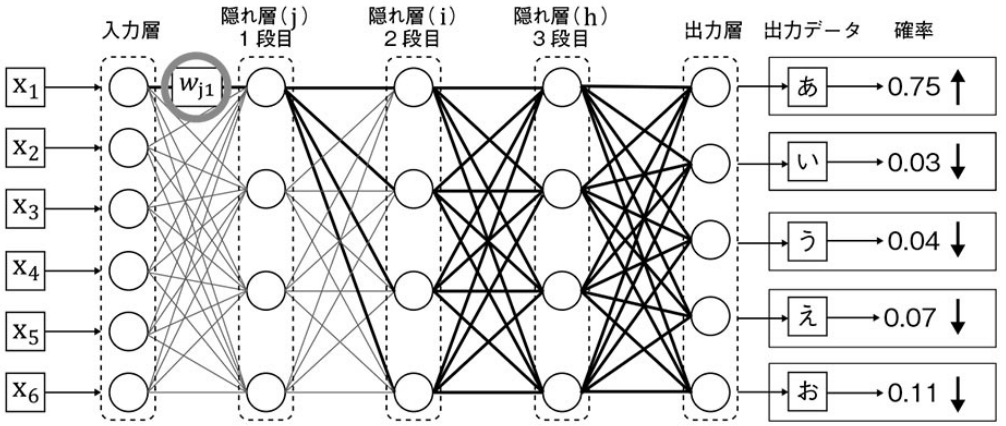


図3-43:誤差逆伝播法の巨視的な理解

たとえば、図3-43のようにそれぞれの層でも複数のノードを持つニューラルネッ

トワークのw_{j1}について重みの更新量を求める際には、出力層から隠れ層1段目に接続される、図の太線で示されるすべてのニューラルネットワークの経路に対して同様の計算を行い、それらについて合計するという作業が必要になります。

図3-43のような数のニューラルネットワークでも人の手で計算するのは大変です。そのため、ディープラーニングで誤差逆伝播法が実装され自動的に計算できる仕組みになったことが、AIの発展に多大なる推進力をもたらしたことを実感いただけるでしょう。

以上が、サンプル数が1個の場合の誤差逆伝播法を用いたすべての層にわたる重みの勾配の求め方となります。

実際のディープラーニングでは、1つのサンプルではなく複数のサンプルを1つの塊として扱うミニバッチの形で学習されることが一般的です。そのため、全データの中からバッチサイズ分のサンプルを取り出し、個々のサンプルに対して損失関数を適用することで得られた誤差を算出し、その誤差の全サンプル分の総和を求め、平均化するという処理を行います。これで重みの更新量が計算できます。具体的な出力層の重みの計算式は、以下のとおりです。

$$\frac{\partial E}{\partial w} = \frac{1}{\mathrm{n}} \sum_{i}^{n} \frac{\partial E_i}{\partial w}$$

数式3-6：出力層の重みの計算式

本節の最後にこれまでの説明をまとめます。

ディープラーニングでは、損失関数を用い出力側の誤差の数値がどのように入力層までの各ノードに影響を与えていくかを調べることで、パラメータの更新の方針が立てられるようになります。偏微分と連鎖律という数学的手法を用いて、出力側から入力側に向かって勾配を計算していくことで、パラメータの更新を実行するというのが、誤差逆伝播法で行っている具体的な処理です。

以上でディープラーニングの学習の動作に関する説明は終わりです。ディープ

ラーニングを実践していくと、勾配消失や勾配爆発などで期待どおりに学習が進まないという事態に頻繁に遭遇します。その際には、本章で説明したイメージを理解しておいたことが助けとなり、チューニング方針を決める際の手掛かりになればと思います。

3.11 まとめ

実際のディープラーニングの処理は、以下のようなプロセスで進めていきます。「3.9.2 データの準備・整理」で説明したデータの準備に始まり、解くべき課題に合わせて、「3.9.3 ディープニューラルネットのモデルの実装」で説明したモデルの形状や活性化関数、損失関数、さらには最適化手法などを適切に設定します。活性化関数や損失関数は、まずサンプルプログラムのコピー&ペーストから始めることになりますが、もう一歩踏み出す際には、本章の「3.9.3 ディープニューラルネットのモデルの実装」、そして「3.4.3 活性化関数とは」「3.4.4 損失関数とは」および「3.5.1 最適化とは」のページが参考になります。

次に、「3.7 ニューラルネットワークの学習の割合や回数に関する指定」で説明したハイパーパラメータを設定し、「3.4.4 損失関数とは」および「3.8 ニューラルネットワークの評価」で説明した評価関数を用いて、誤差を評価し誤差が最小になるまでパラメータを更新していくという作業を行うことになります。

実際に更新作業を進めていく際に、学習が進まないなどの状態が生じた場合には、さまざまな試行錯誤を繰り返すことになりますが、そのような状態に陥ったら、本章の「3.6 ニューラルネットワークの隠れ層の学習についての直感的理解」「3.8 ニューラルネットワークの評価」「3.10.1 誤差逆伝播法とは」の内容が特に参考になるでしょう。

本章で基礎的な理解はできたはずなので、次の章はアドバンスユーザ向けに本格的なディープラーニングの洞察につながる情報を提供します。

第4章

畳み込みニューラルネットワークのメカニズムと意味をとらえる

フレームワークの使用方法を超えた知識の必要性

Deep Learning Project Template

最初に本章の目的について説明します。Kerasのような標準的なディープラーニングフレームワークの近年の進化により、機械学習や数学的な感覚についてのベースの知識を十分に有しない多くの人でも、ディープラーニングのモデルを簡単に構築する機会が増えてきています。そのような人たちにディープラーニングのメカニズムについて基本的な質問をしたときには、非常に限定的な考察力に基づくために、自分たちが何を構築しているのか、またその構築理由がどういうものなのかが十分に理解できていないと思われる場面に遭遇します。特に実験のサイクルを回していくときに、常に仮説を立てながらモデルを構築していく必要があります。そうした機械学習の技術的な特性により、これらの技術のメカニズムを理解することが、エンジニアにとってソリューションを正しく設計するのに役立ちます。より正しい仮説を構築するには、それぞれのコンポーネントがどのような役割を果たしているのかを理解することが重要です。そのため本章では、**畳み込みニューラルネットワーク**（Convolutional Neural Network：以降、**CNN**）で用いられる主要なコンポーネントに関する役割とその意味についての理解を深めることに注力します。

4.1 CNNの本質を理解するために

▶4.1.1 立ちはだかる学習リソースとのギャップ

CNNについて、必要となる他の学習リソースを参考にした場合、チュートリアル

は主に単純な操作の概要に焦点を当てていて、通常それらの操作の意味の解釈については説明していません。一方で、顧客に満足がいくサービス提供が求められる商用という厳しいエンタープライズの環境でアプリケーションを構築する場合、問題はしばしば固有のものであり、使用される環境ごとに異なります。したがって、基本的なチュートリアルから固有の問題に知識を適用しようとする際には、一般的に大きなギャップがあります。このギャップを埋めるために、本書では、CNNがどのようなものであるかを単なる（数学的な）操作で説明するのではなく、その導入部分の説明は手短に留めて、可能な限りCNNの解釈を示すようにします。また、CNNが画像などの視覚的な特徴を抽出したり、パターンを認識したりするのに優れている理由に加えて、動作の根幹をなす畳み込み演算について学びます。この演算を知ることにより、特定の問題に対して最適なモデルを見つけるために、実験サイクルを正しく繰り返していけるようになるでしょう。

▶4.1.1.1 ギャップを埋めるためのチャレンジ

では、基本的なチュートリアルと固有の問題に存在するギャップをどうすれば埋めることができるでしょうか。

私たちは、「このテクノロジーの本質を直感的に理解すること」から出発することだと信じています。そのため、本章では、テクノロジーの主要なコンセプトについて直感的に説明していくことを通じて、AIのコミュニティがさらに研究を続け、テクノロジーの将来に貢献できるような情報提供を進めていきます。本章は、次の3つのテーマについて説明します。

1. 本テクノロジーの本質の直感的な理解
2. 既存のリソースに欠けていると思われるトピックスの紹介
3. しばしば混乱を引き起こす概念に関する説明

◉――本書で取り扱わない内容

ディープラーニングの分野は、日々新しいテクノロジーが生まれ急速に発展しているため、研究内容によっては数週間ごとに新しい最高のアーキテクチャが明らかになっています。そのような状況において、現在の最高のアーキテクチャまたは最先端のアプローチを提示することは不可能です。そのため、本書は最先端のアーキテクチャを紹介することを目的としてはいません。また、他の文献やネットなどで容易に見つけることができる情報については、本質を理解するために必要な知識を中心に、最小限に抑えるよう努めています。その代わりに、本章では、CNNの動作の本質についての解釈をできるだけ示します。CNNというアーキテクチャの根幹にある畳み込み演算からインスピレーションを得て、CNNが視覚的特徴の抽出とパターン認識に優れている理由を説明します。それらを知ることによって、実験のサイクルをさらに適切に繰り返せるようになり、読者特有の問題に対しても最も適したモデルを見つけられるような知識を得られるはずです。本章の以降の内容は以下のように構成されています。

4.2 CNNの概要
- 4.2.1 CNN誕生の背景
- 4.2.2 全結合型と畳み込みの違い
- 4.2.3 画像の入力データ
- 4.2.4 CNNの構成要素と操作

4.3 CNNの特徴
- 4.3.1 カーネルの役割
- 4.3.2 CNNにとっての学習とは
- 4.3.3 畳み込みの階層構造
- 4.3.4 パラメータ共有
- 4.3.5 受容野
- 4.3.6 パディング

4.4 まとめ

それではさっそくCNNの世界を見ていきましょう。

4.2 CNNの概要

本節では、まずCNNの概要を紹介します。

◉── CNNがAI業界のアプリケーションに与える多大なる影響

CNNは現在、コンピュータビジョン（画像処理）のさまざまなタスクで最も広く利用されているニューラルネットワークアーキテクチャです。CNNが使用されている具体的な分野としては、画像内の物体の認識を行う画像認識、画像に含まれる任意の数の物体を検出する物体検出、画像の領域分割などがあります。さらに近年では、自然言語処理の分野でもテキスト分類や感情分析など幅広く利用されるようになっています。実際に皆さんが利用されているサービスでも、Googleの写真検索やLINEのリコメンデーションなどさまざまなものに利用されています。

CNNが現在のAI業界のアプリケーションにもたらしている貢献は絶大であり、CNNに関する基本的な情報を見つけるのは難しくありません。ただ前述したように、基本的な知識と実際に利用できるようになるまでの間にはギャップが存在します。本章では、そのギャップを埋めるために、数学的な操作を用いてCNNがどのようなものであるかを単に説明するのではなく、数学的な導入部分はできるだけ端的に説明し、CNNの動作の本質を中心に説明していきます。

CNNに関するネット情報や関連本の出版数が増えていることからも、CNNが非常によく利用されていることについては理解されているものと思います。ただし、CNNで使用されているテクノロジーをきちんと理解して解釈するには不十分なケースが多く、入門書で得られる知識と企業などで実際のアプリケーションを実現しようとすることの間には大きなギャップが存在します。本節では、CNNが必要

となった理由とCNNを利用する利点を中心に説明をしていくことで、そのギャップを埋めていきます。

▶4.2.1 CNN 誕生の背景

画像処理のタスクについて、**全結合型ニューラルネットワーク**を使用して学習を行う場合、モデルは画像の視覚表現のすべてのパターンにあたるすべての画素値（ピクセル値）の組み合わせを学習する必要があります。ここでは、画像データベースの1つであるMNISTの例を用いて説明をしていきます。

▶4.2.1.1 全結合型ニューラルネットワークから学ぶ CNN の必要性

まず、分析を行うデータについて説明します。画像に関する入力データの種類としては、さまざまなものが存在します。データベースは、Excelの表形式のデータに代表されるようにカラム形式のデータを扱いますが、画像分析では、画像データは以下のように、それぞれの画素（ピクセル）の色の濃淡を数値で示した縦横の行列で構成されています。図4-1はMNISTにある手書き数字の画像データの一例で、数字5の手書きデータになっています。

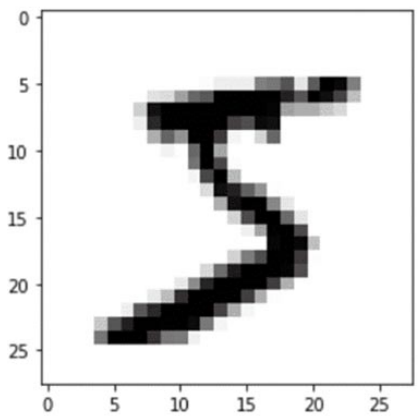


図4-1：白黒画像の例

また図4-2は、例の手書きデータの具体的なデジタル表現となります。

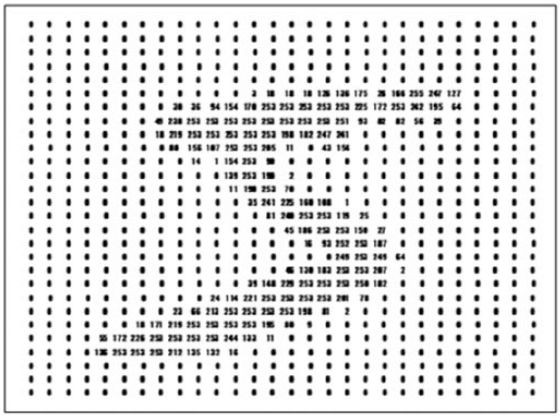
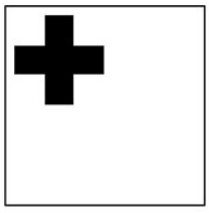


図4-2:前例の画像データのデジタル表現（各画素値）

上記のような画像データを全結合型ニューラルネットワークで学習する場合、どのような問題があるでしょうか。それは、物体認識において画像内に写っている物体の位置の問題です。

たとえば、画像データの物体検知問題を解くタスクでは、左側に物体があっても右側にあってもそれを正しく検知する必要がありますし、物体が小さめに写っている写真であっても大きめに写っている写真であっても同じ物体であることを検知する必要があります。

説明を簡単にするために、図4-3のように6×6の画素で構成された画像があるとします。それぞれの画素は白が0、黒が1と表現されるものとし、画像には+の物体が写っているとします。



＋の画像

0	1	0	0	0	0
1	1	1	0	0	0
0	1	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

＋の画像のデジタル表現
（注：簡素化のため各画素は、白黒の 0,1 の数字で表現）

図4-3:＋の画像と、画像のデジタル表現のイメージ

上記の写真で何が写っているかを検出する、といった物体検出の問題に対し

て、全結合型ニューラルネットワークを用いてみます。

全結合型ニューラルネットワークで上記の画像データを学習させる場合には、下の図4-4のように、入力の画像データをフラットな1列のベクトルに変更することから始める必要があります。

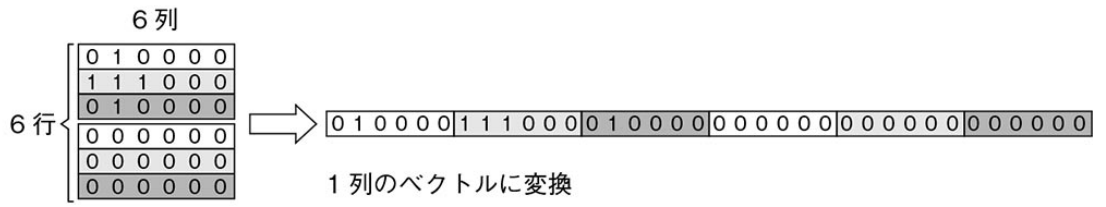


図4-4：画像データの全結合型ニューラルネットワークへの入力ベクトル変換

フラットな1列のベクトルが準備できたところで、全結合型ニューラルネットワークでは、特徴を学習できるように接続します。特徴を学習するというのは、1つ1つの入力値であるピクセル値に対して、図4-5のようにそれぞれの値の程度によって出力がどうなるか、といったことです。

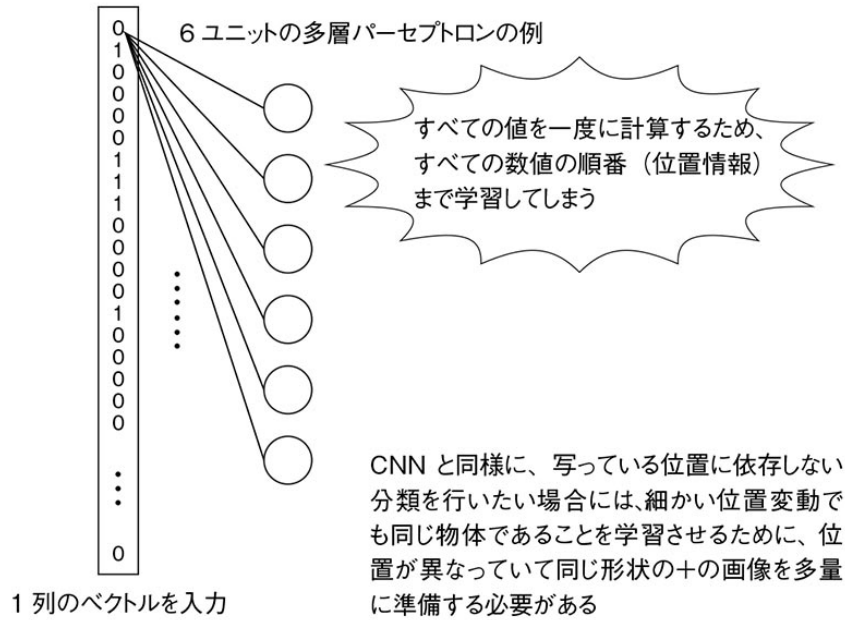
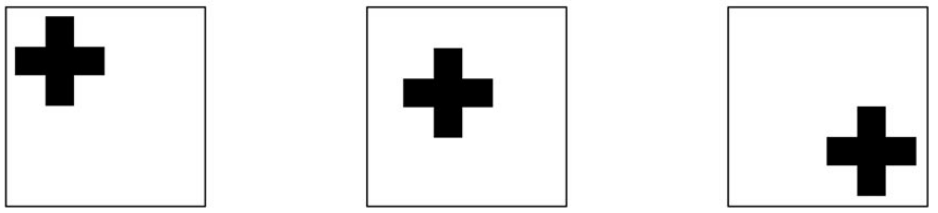


図4-5：全結合型ニューラルネットワークを用いた画像データの学習イメージ

このようなやり方の問題点は、図4-5のような1列のベクトルの上から下まですべての画素がネットワークに全結合で接続されているため、画像に写っている物体の位置に相当するそれぞれのピクセルの位置についても基本的に学習していることです。つまり、少しでも入力データの位置やサイズが違う画像データがあった場合には、同一物体であっても正しく検知できなくなってしまうことになります。仮に物体の全結合型ニューラルネットワークで特定の位置によらないモデルを作るには、同一の物体に対して位置や大きさが異なる膨大な量の写真を準備して学習させる必要があります。このような学習は理論的には可能ですが、正しく学習させるために膨大なデータを準備するのに労力がかかりすぎるため、現実的には難しい方法となっています。

もう少し具体的に説明しましょう。図4-6はいずれも+の画像ですが、+の位置が左上か真ん中か右下かという違いのみが存在するものです。たとえばそれぞれの画像に写っているものが何なのかを特定する物体検知のような問題を解く場合には、それぞれの画像の絶対的な位置は問題にならず、写っているそれぞれの物体の特徴のみを知る必要があります。その点で基本的に写真の中の物体の位置まで学習してしまう全結合型ニューラルネットワークでは、物体の撮影位置に依存しないということ自体を学ばせるために、位置が異なる同一画像を多数用意して学習させる必要が生じてしまいます。



＋の画像　　＋の画像　　＋の画像

＋という物体を検知する場合、写真の左上、中央、右下にあっても＋という画像であることに変わりがないため、撮影された絶対的な位置を知ることには意味がない

図4-6:＋の画像例

人が写真から物体を検知する場合には、位置や大きさは気にせず物体を正しく検知できます。これと同じ機能を実現するために、ディープラーニングにおいても写真の位置、大きさ、細かい特徴の違いなどに左右されない方法として、全結合型ニューラルネットワークよりも、学習効率や精度が良い仕組みが必要になります。その仕組みを実現するために考案されたディープラーニングのアーキテクチャが、本章で説明するCNNです。

▶4.2.2 全結合型と畳み込みの違い

図4-7に、全結合型ニューラルネットワークと畳み込みニューラルネットワークの学習の違いについて概略を示します。

図の左の全結合型ニューラルネットワークは、すべての入出力が結合され、それぞれに固有の重みを用いて画像に含まれている特徴を獲得していくため、画像の位置情報のようなものも含めて学習しやすい傾向にあります。これまで説明したように、物体検出のような物体の写っている位置や角度によらない解析を行いたい場合には、不向きなアーキテクチャだと理解できます。また、重みはすべての入出力に対して基本的には異なり、パラメータ数が非常に多くなるため、計算コストが高く、コンピュータのリソースが必要になるというデメリットもあります。

それに対し、畳み込みニューラルネットワークは、画像中の一部分の特徴を学習するために準備された**カーネル**と呼ばれる、入力画像データより小さい行列を用います。画像の一部分の特徴を学習するということを画像全体にわたって行うため、物体の位置が異なる画像であっても同じ物体の特徴を示す画像が存在した場合には、的確に物体の検出が行える仕組みになっています。これらの詳しい説明については、後述します。また、CNNの畳み込み層では、カーネル（フィルタ）が全結合型ニューラルネットワークの重みにあたる役割を果たすため、パラメータ数がカーネルサイズ×カーネル数になり、全結合型ニューラルネットワークより少なくて済むというメリットもあります。

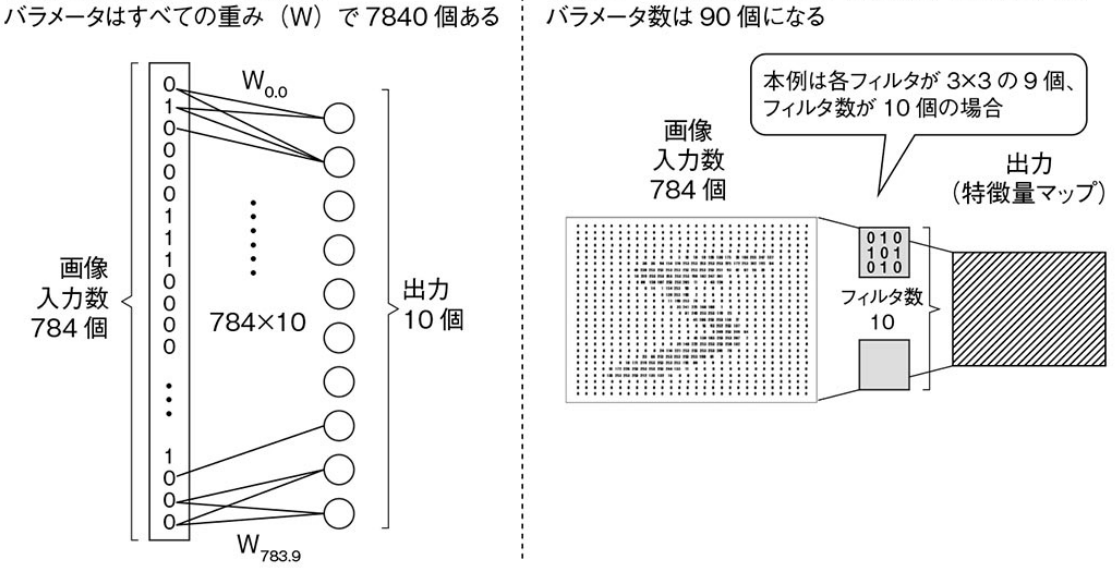


図4-7：全結合型ニューラルネットワークとCNNの学習の違い

CNNに限ったことではありませんが、画像処理では、入力データにあたる画像データについての知識が必要となります。そこでCNNの本題である仕組みを説明する前に、入力データの基礎知識として、CNNに取り込む入力画像データはどのようなものなのかを簡単に説明します。

▶4.2.3 画像の入力データ

読者の皆さんはすでにご存じかもしれませんが、1枚の画像データがグレースケール（白黒）画像の場合、一般的に、各ピクセルが0から255までの範囲の値を示すピクセル値の集合で表現されます。画像データでは、図4-8のように水平方向に並んだピクセル数は画像の幅（Width）と呼ばれ、垂直方法のピクセル数は高さ（Height）と呼ばれます。CNNで用いる画像は大別すると、グレースケール（白黒）画像とカラー画像の2種類が存在します。グレースケール画像は、各ピクセル値の色の濃さ（強度）のみで表現され、黒／灰色／白のグラデーションとなります。カラー画像は、色の表現を実現するために、次の図で示される、いわゆる「チャ

ネル」を使用して表現されます。具体的には、カラー画像には3つのチャネルがあり、各チャネルは赤、緑、青色がそれぞれ割り当てられていて、それぞれの色のグラデーションの組み合わせで色が表現されます。たとえば、赤チャネルの各ピクセル値は、赤の色の強度を表し、緑と青色についても同様になります。学生時代に光の三原色でも習ったと思いますが、三原色の組み合わせを用いることで、さまざまな色を表現することが可能になります。

▶4.2.4 CNNの構成要素と操作

入力の画像データについて理解できたところで、ここからはCNNのメインテーマである、CNNの構成要素とその操作について説明します。

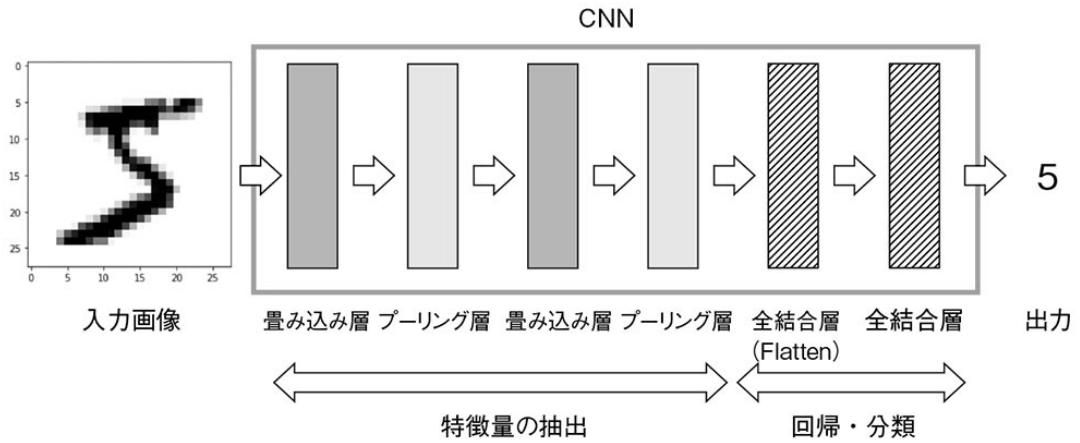


図4-8：基本的なCNNの構成イメージ

図4-8の太枠の四角の中のように、CNNは畳み込み層、プーリング層、全結合層で構成されます。それぞれの層の役割は以下のように要約できます。

- 畳み込み層：入力データに対して特定の特徴量を抽出する役割
- プーリング層：畳み込み層で得られた特徴量マップ（前の層から出力された特徴量の全体）をダウンサイジングする（解像度を落とす）役割
- 全結合層：前の層で得られた特徴量マップを1次元のベクトルに変換した後、解く

べき問題に従い分類（または回帰）を行う役割

実際のCNNでは、入力から出力までの間に、のちほど詳しく説明する畳み込み処理とプーリング処理が複数回用いられます。それぞれの畳み込み層の入力には、入力画像もしくは前段の畳み込み処理の出力となる特徴量マップが用いられます。畳み込み層における畳み込みの計算は、学習可能な重みを有する固定サイズのフィルタ（カーネル）を入力データ上にずらしていくことで実行され、その出力は入力のデータと同等かそれより小さいサイズの特徴量マップになります。図4-8のように層を重ねていくのには、理由があります。CNNは画像に限らずさまざまな特徴量を抽出可能ですが、画像データから特徴量を抽出するという文脈でとらえた場合、各層の学習された特徴量マップは、最初の層では画像の低レベルの特徴をとらえ、層が深くなるにつれ、物体などのより細かく、かつ全体的な特徴をとらえるようにする必要があるからです。

プーリング層では、畳み込み層で学習した特徴が位置ずれを起こしても正しく認識できるように、特徴量マップをダウンサイジングします。最後の全結合層では、出力層で求められる分類を実行できるように、入力の特徴量マップをフラットな1次元のベクトルに変換し、実際の分類を行います。個々の機能についての詳しい説明は、のちほど紹介しますが、以上がCNNの概要です。

実際のCNNでは、解くべき問題によって畳み込み層やプーリング層が3層程度のものから、数百層になるものまで、その構成はさまざまです。ただ基本的な操作は変わりません。そこで以降では、畳み込み層、プーリング層、全結合層の役割をそれぞれ詳しく説明していきます。

▶4.2.4.1 畳み込み層

最初に**畳み込み層**の役割とその動作について説明します。図4-9は、カラー画像（紙面ではグレー）を入力データとして用いた場合の畳み込み処理の流れになります。左がカラー画像の入力データで、RGBの3チャネルのデータとなっています。

次のフィルタを用いて畳み込みの計算が実行され、その計算結果が出力層に送られて特徴量マップを形成します。

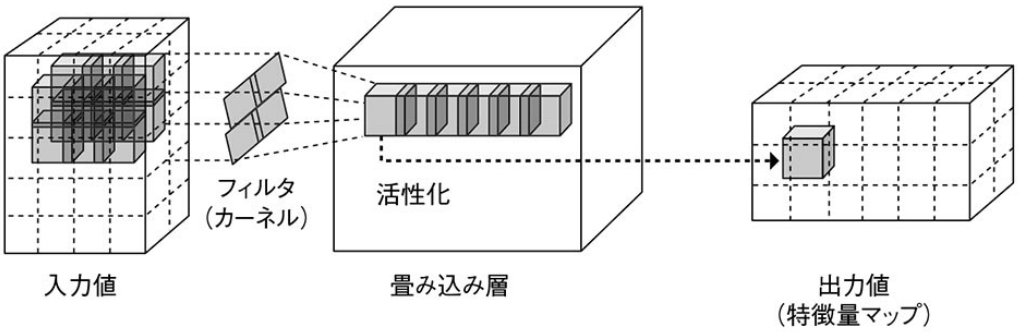


図4-9：畳み込み層の実装イメージ

これから畳み込みの計算を具体的に説明します。説明をわかりやすくするために、図4-10の例では、左の5行5列の行列が入力画像を意味します。3チャネルをそのまま説明すると複雑になるので、ここでは1チャネルに限定して説明します。3チャネルは、1チャネルでの操作と本質的には同じなので、そのような説明で問題はないはずです。また、入力画像のピクセル値は、0か1のいずれかで構成されるものとします。図の中央の上の3行3列の行列はカーネル（もしくはフィルタ）を意味します。先の項で説明した全結合型ニューラルネットワークでは重みをもとに特徴量を計算しましたが、畳み込み層では、カーネル（もしくはフィルタ）が重みの役割を果たします。

では、具体的な畳み込みの計算を行います。実は非常に簡単です。まず図の左の画像では、行列の左上の3行3列からなる画像の各要素に対してカーネルの各要素を掛けます。図の中央下の3行3列の行列では、実際に同じ位置にあるそれぞれの要素同士を掛け合わせています。次にこれらの要素の計算結果を合計します。これだけです。

この例の場合、合計値は図の右の行列の一番左上にある値です。

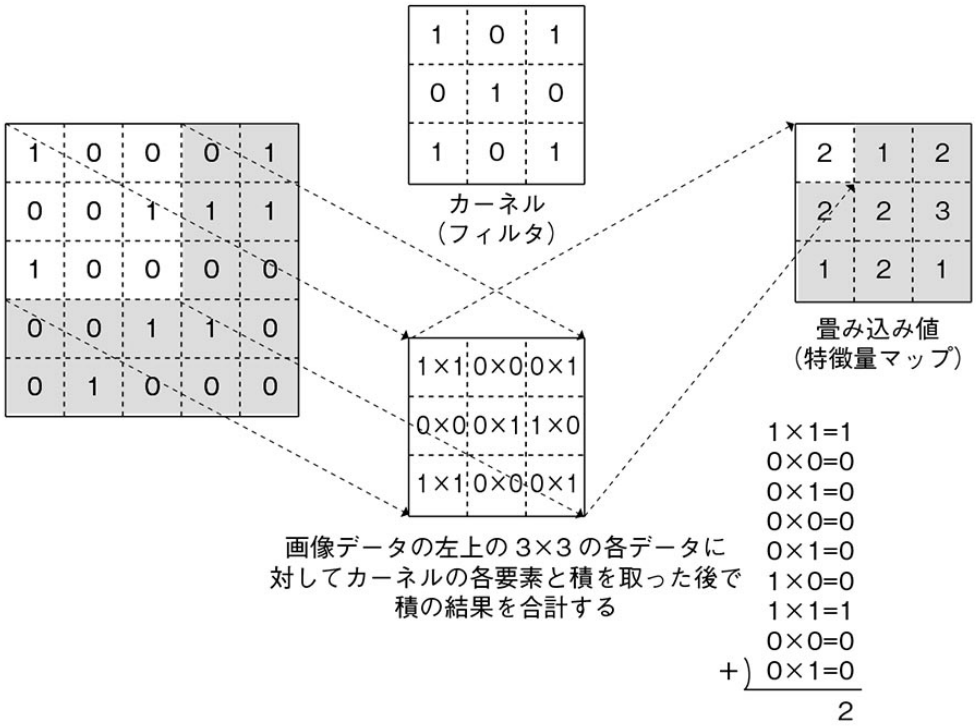


図4-10: 畳み込み計算処理例

1つのカーネルによって画像全体に対して畳み込み処理を行うには、カーネルを入力画像に対して1ピクセルずつずらして（スライドして）畳み込み計算を実行します。最終的に図4-10の右にあるような畳み込み値の集合が出来上がります。この畳み込み値の集合は**特徴量マップ**と呼ばれます。

◉――カーネル

ここで**カーネル（フィルタ）**について説明します。ディープラーニングの文献では、カーネルとフィルタはしばしば同じ意味で使われていますが、正確に言えばフィルタはいくつものカーネルが集まったカーネルの集合ということになります。そのため、2次元の重み配列はカーネルと呼ばれ、3次元の重みのボリュームは2次元の重み配列が3チャネル集まったものなので、フィルタと呼ばれます。ただし、基本的な畳み込みの操作という観点では、カーネルもフィルタも同じ意味で使用されることが多いため、ここではカーネルとフィルタは同じ意味だと考えて問題ありません。具体的な畳み込みの処理を見ていくために、以下の例では、3行3列のカーネル

を用いたストライドの操作について説明します。

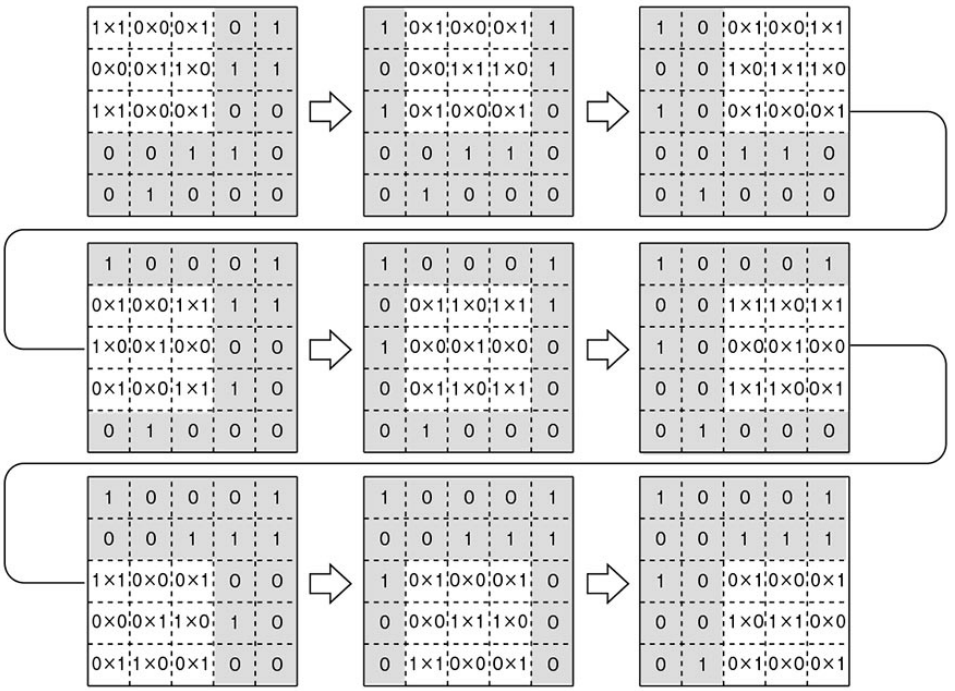


図4-11：畳み込み処理のストライドの例

図4-11を見てわかるとおり、入力データに対して、カーネルが縦横1ピクセルずつ隣に移動（ストライド）し、そこで畳み込みを計算することにより、カーネルを介した特徴量マップが抽出されます。画像の左上の角から畳み込みの計算を開始し、右方向に1ピクセルずつストライドして畳み込み計算を行っていき、右端までたどり着いたら、1ピクセル下にずらしてまた左端から計算を行っていくような処理を繰り返すことで、全画像に対して畳み込みが実行できます。

▶4.2.4.2　プーリング（Pooling）層

次に、**プーリング**の役割とその動作について説明します。

画像データを扱うには、画像に写っている物体の位置がずれることによって物体の認識率が下がらないようにするだけでは不十分で、その物体の回転や、大小つまり画像の拡大・縮小に対しても物体の認識率が下がらないようにしなくては

なりません。このような回転や拡大・縮小に対応する仕組みがこれから紹介するプーリングです。プーリングを畳み込み層の後に実装することで、畳み込み層で浮き出てきた特徴をぼかすことができます。これにより、それぞれの画像に特有な物体の回転や大小などの細部にわたる違いに左右されずに、大まかな特徴をとらえられるようになります。

物体検知のような問題では、同一の物体であっても、画像内の物体の位置が少しずれていたり、見える角度が変わったりすると、少し見える形が違ったりします。そのような小さな変動まで学習しないようにする仕組みが必要で、プーリングがその機能を果たします。このように小さな変動や移動を学習しない仕組みは**移動不変**(translation invariant)と呼ばれます。

プーリングを使用することにより、入力画像の小さな変換に対して不変性が与えられます。これに加えて、画像上の位置に左右されないように物体検出などの問題を解く場合に、より優れた精度のモデルを作成できるようになります。その理由をイメージできるように、以下ではプーリングの説明を進めていきます。具体的な説明に入る前に、プーリングを用いるタイミングとプーリングのサイズについては注意が必要なので、ここで簡単に説明します。プーリングは、特定のサイズのピクセル領域周囲の情報を集約する機能であるため、入力画像内の物体の正確な位置に関する情報が失われてしまいます。物体検出のモデルなどでは物体の領域を精度良く定める必要がありますが、プーリング操作による出力では、正確な位置情報自体が失われてしまいます。その場合は、実際に解決したい課題に合わせて適切なプーリングの実装位置を検討する必要があります。

これからプーリングの機能と操作の概要について、さらにはプーリングの代表例について説明していきます。

▶4.2.4.3 プーリングの役割

プーリングには、主に以下の2つの役割があります。まず、それぞれの役割を説明します。

●画像のデータ量削減

●移動不変性

◉――画像のデータ量削減

まず、画像のデータ量の削減について説明します。図4-12の左側のように、15×15の画像があったとします。代表的なプーリングの方法は2種類ありますが、ここでは一般的に用いられる**Max Pooling**を説明します。Max Poolingの場合、入力画像データに対して3×3の行列内の最大値を返すような処理を実行した結果は、図の右のような出力になります。これは元の225ビット（15×15）の画素の画像が25ビット（5×5）にまで圧縮されたことを意味します。つまり、プーリングの操作には画質を落とす効果があり、細かい特徴は失われますが、大まかで重要な特徴は失われることなく出力されることを意味しています。

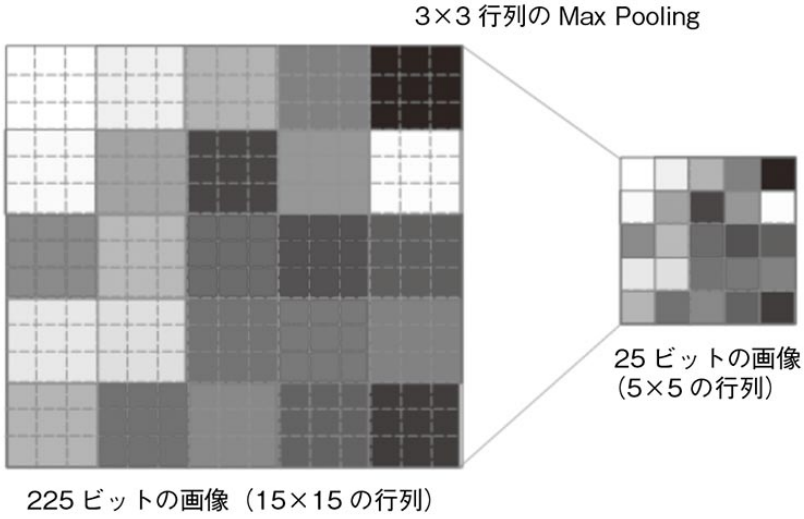
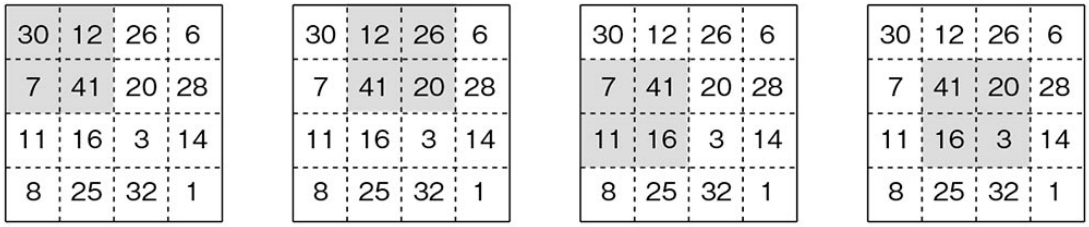


図4-12:画像のデータ量削減のイメージ

◉――移動不変性

次に、プーリングがもたらす移動不変性（translation invariant）の効果について説明します。図4-13を見てください。左の画像が元の画像だとして、左から2番目は、1ピクセル分グレーの部分を右にずらした画像です。右から2番目の画像

は1ピクセル分を下にずらした画像であり、一番右の画像は右と下に1ピクセルずつずらした画像です。いずれの画像でも、グレー部分で示されている2行2列の行列の部分に対してMax Poolingを実行すると、その計算結果はすべて41となります。このことはMax Poolingを用いたことにより、多少の位置のずれを起こしても出力には影響を及ぼさない、つまりMax Poolingを使用しないモデルよりも位置ずれに左右されない物体検知が行えるモデルになるということを示しています。今回の例は上下左右の位置ずれの例でしたが、同様のことが画像の回転に対しても言えます。プーリングを用いることで、上下左右や回転のような位置ずれに対して不変的な出力を返せるようになるのです。



元の画像の行列から 1 ピクセルずつずらして 2×2 行列の Max Pooling を実行しても
すべて 41 という値が得られる。すなわち小さな位置変動に対して出力が不変になる

図4-13:移動不変性の例

◉――プーリングの操作

プーリングの操作自体も非常にシンプルです。図4-14の例を見てください。図4-14の左のような4行4列の行列があるとします。

最初にMax Poolingの例について説明します。4行4列の行列のうち、左上のプーリングのサイズが2行2列の行列に対してMax Poolingを行う場合を考えます。当該行列にMax Poolingの操作を行うことで、各要素のうち最大値の41が得られます。つまり、今回の例では、2行2列の行列に対してMax Poolingを実行することにより、その要素の中の最大値の1つを出力する、つまり入力行列の4つの要素を最大値という1つの出力の要素に圧縮します。言い換えると、プーリングの対象となるデータ領域の中から、1つの代表する特徴量として、最大値を選択

することになります。

次に、**Average Pooling**について説明します。Max PoolingとAverage Poolingの違いは、Max Poolingは行列要素の中の最大値を選択するのに対し、Average Poolingは各行列要素の平均値を出力することです。Max Poolingとプーリングサイズが同じ2行2列として、図4-14の左上の2行2列の行列に注目してみます。Average Poolingでは、4つの行列要素である30、12、41、7を合計し、行列要素数の4で割ることで各要素の平均値22.5を算出します。

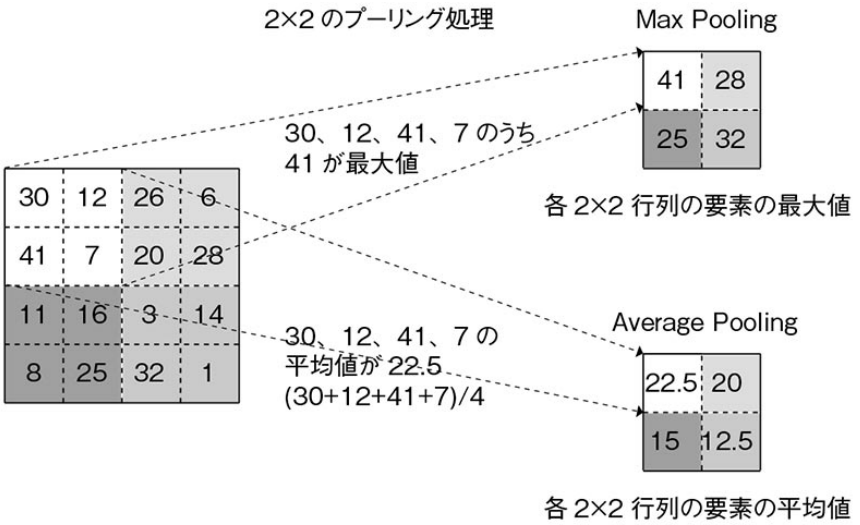


図4-14:プーリングの種類

どちらのプーリングを用いるかは解こうとしている課題によりますが、一般的には、画像の位置が重要ではなく高速処理が必要な場合にMax Poolingを用いるので、まずはMax Poolingで試すことをお勧めします。

◉――グローバルプーリング

近年のCNNでは、回帰や分類を実行する層では、全結合型ニューラルネットワークに代わり、以降で紹介する**グローバルプーリング**が用いられる傾向にあります。

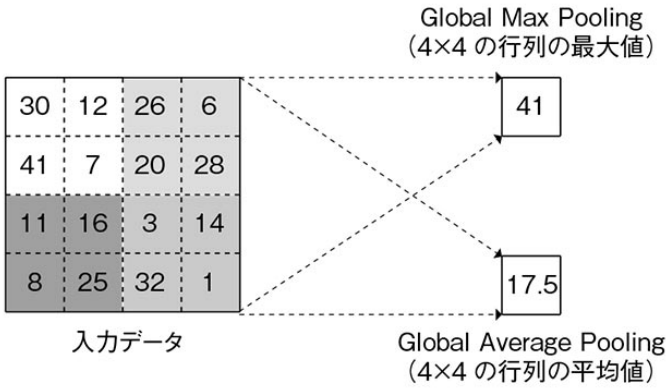


図4-15:グローバルプーリングの種類

グローバルプーリングは、プーリングのサイズが入力データのフルサイズになるだけです。計算自体はプーリングで示したものと同じで、Global Max PoolingとGlobal Average Poolingが存在します。

主に、これらのグローバルプーリングを使用して、畳み込み層によって出力された特徴量マップの次元を削減できるので、全結合層のFlatten層（複数次元のデータを1次元のデータに変換する役割を果たす）や、場合によっては出力層の分類器で用いられるDense層（全結合層）を置き換えることができます。

入力データのサイズが大きい場合、Max Poolingを用いると特異な外れ値などが含まれてしまい、精度が落ちる可能性があります。そのため、グローバルプーリングではGlobal Average Poolingを用いるのが一般的です。

▶4.2.4.4 CNNにおける全結合層

ここまで見てきた畳み込みやプーリングでは、出力がいずれも特徴量マップ、つまり画像形式の配列で構成されています。

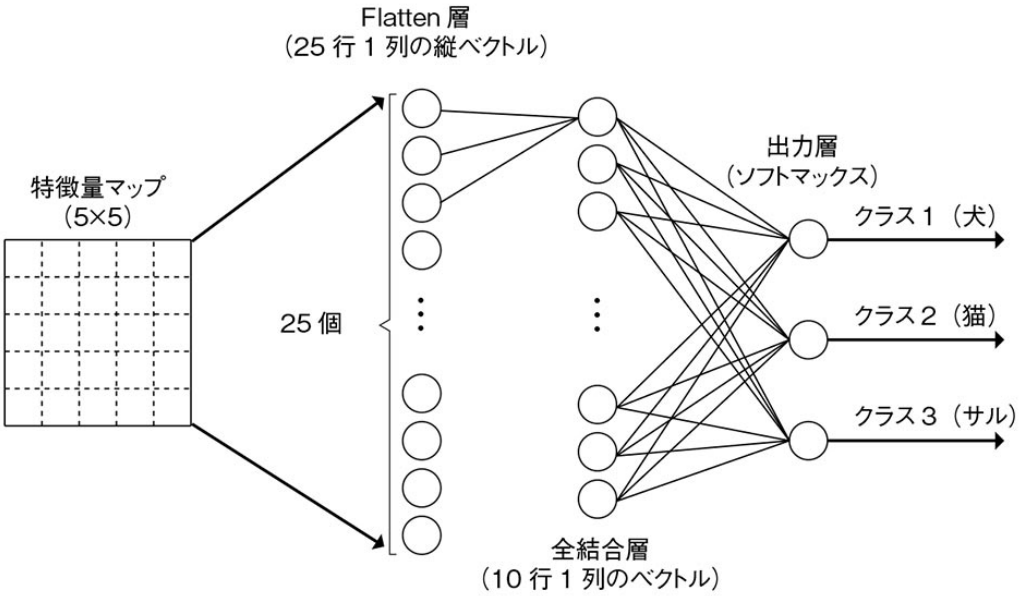


図4-16:CNNにおける全結合層の役割（フラット化と分類）

入力画像から「犬」や「猫」などの物体を分類するには、CNNの処理のどこかで画像の配列形式から、分類処理が行いやすいデータに変換する必要があります。そのため、CNNでは、畳み込みとプーリングを何層か実行した後、画像データを1列のベクトルに変換するフラット化の処理を行います。

具体的に、図4-16のような出力層の直前の5×5の特徴量マップに対して、フラット化の処理を行い、分類結果を出力するまでを説明します。

まず、図の左の5×5の特徴量マップを、Flatten層（フラット層）を用いて1列のベクトルに変換します。図の例では、5×5の2次元行列が25行1列のベクトルに変換されています。

一度、フラットな形式のデータに変換されてしまえば、先に説明した基本的な全結合型ニューラルネットワークの仕組みがそのまま使えるようになります。その後は、分類問題を解くときに、ソフトマックス関数などによる分類が可能となります。図の例では、10ノードの全結合層を用いた後、多クラスの分類であるため、活性化関数にソフトマックスを用いることで、犬、猫、サルなどの3つのクラスに分類しています。

CNNの構成要素とその役割の説明は以上です。CNNは「畳み込み層」「プーリング層」「全結合層」が組み合わされた構成となっていて、それぞれが「特徴量の抽出」「ダウンサイジング」「フラット化と分類」という役割を担っていることが理解できたと思います。

今までのところで基本的な概念は説明しました。次の節では、CNNで使用される主な機能の役割について掘り下げていくことで、各機能の本質についての理解を深めていきます。

4.3 CNNの特徴

本節では、CNNが画像の特徴となるパターンをどのように認識しているのかを理解するために、重要なそれぞれの項目について詳しく見ていきます。そうすることでCNNの動作の本質について理解を深めることができます。

▶4.3.1 カーネルの役割

本章で先に説明したように、ディープラーニングでは重みを用いて入力データに含まれる特徴を学習していきます。CNNでは、重みに相当する機能として、カーネルが特徴量抽出の役割を果たします。カーネルが特徴量抽出の役割を果たすことを実感できるように、以下の例で説明します。

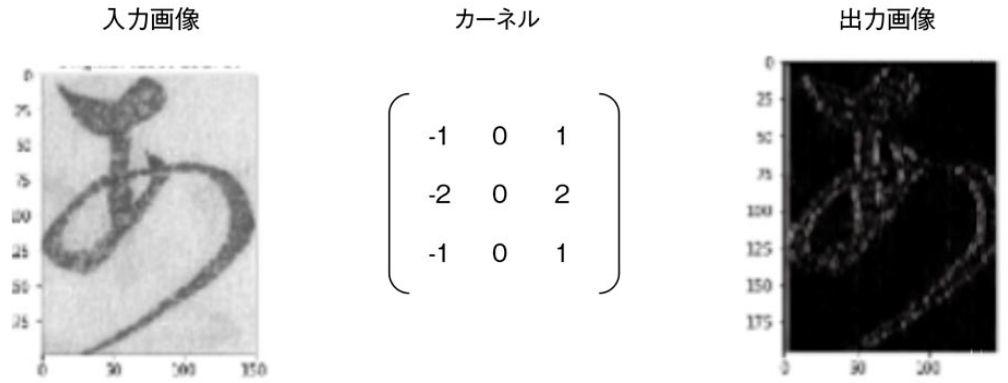


図4-17:エッジを検出するカーネルを用いた画像出力例

図4-17は、MNISTの崩し字の「あ」という文字に対して、図の中央にある3行3列のカーネルを用いて畳み込みを行い、得られた結果が特徴量マップとして右の図に示されています。この中央のカーネルでは、左1列はすべてマイナスの値、中央の列が0、右1列が正の値になっています。これは、縦の0の領域を挟んで左右で数値が異なる、つまり物体の縦の境界のような、数値変動が激しい領域の特徴をとらえるカーネルとなっています。そのため、右の「あ」の特徴量マップを見るとわかるとおり、縦線のコントラストが強い「あ」という文字の輪郭の部分だけが強調されて白抜きされています。

実際のCNNのカーネルは、全結合型ニューラルネットワークの重みにあたるものなので、学習が進むにつれて調整されるパラメータです。ただしここでは、カーネルがどのような特徴を抽出するのかを理解できるようにするため、以下で典型的な特徴を持つカーネルの例を紹介します。

図4-18の左のカーネルは、図4-17のカーネルの縦横を変えたものなので、横のエッジが検出されます。中央のカーネルは、真ん中のピクセルだけが強調されるため、画像のよりシャープな特徴が検出されます。右のカーネルは、中央のピクセルだけでなく、その周辺もすべて同等に扱うため、ぼかし的な効果が得られます。

-1	-2	-1
0	0	0
1	2	1

横のエッジ

0	-1	0
-1	5	-1
0	-1	0

シャープ

1	1	1
1	1	1
1	1	1

ぼかし

図4-18:3つのカーネルの例

▶4.3.2 CNNにとっての学習とは

CNNにとって、カーネルは全結合型ニューラルネットワークの重みの役割を果たすパラメータであるため、学習フェーズでカーネルの値は固定されていません。

実際、カーネルのそれぞれのピクセル値は学習プロセスの中で、調整されていきます。特定のタスクに対してベストな結果を得るために（たとえば分類問題において入力画像を正しく分類するために）、CNNで最適化を行うには、定義された損失関数を最小化する必要があります。この最適化の過程で、カーネルのそれぞれのピクセル値の組み合わせを学習することにより、最適な特徴が抽出されることになります。たとえば画像に猫が含まれているか否かを判別しようとする場合、猫の特徴である耳の形やひげなどの特徴をできるだけ効率的に高い精度で判別できるように、カーネルを最適なものに調整していきます。これがまさにCNNの学習であり、そこでは特定の目標を達成するために、潜在的な特徴を抽出することを学んでいます。さらに、一般的なCNNのアーキテクチャでは、多数の異なるカーネルが適用され最適化されていきます。

以上の説明から理解できるように、カーネルは、畳み込み演算の中心をなす構成要素の1つです。実際のCNNでは、複数の層に対して畳み込みが適用されています。次に、この多層構造が特徴量抽出の観点でどのような意味をなすのか、その意味を探っていきます。

▶4.3.3 畳み込みの階層構造

ここからは畳み込み層を階層構造にする理由を見ていきます。全結合型ニューラルネットにおける多層化で説明したように、層が増えることで入力データに近い層では単純な特徴をとらえ、出力層に近くなるにつれ、複雑なパターンを抽出できるようになります。CNNでも状況は同じです。

つまり、入力層に近いCNNの層では、境界や縦線などの単純な特徴が検出できます。それに対し、出力に近い層では、たとえば動物などの場合には目や耳などより広範囲の特徴が検出できます。これはCNNで階層構造を実装することにより、物体の構造的な特徴を把握できるようになることを意味します。つまり、層が深くなる（出力層に近くなる）につれ、単純な特徴量を組み合わせた構造を有する情報が形成されていくことを示しています。言い換えると、CNNのニューラルネッ

トワークがある程度深くなってくると、目的となる特徴量をとらえるのに必要十分な特徴の抽象化および汎用化ができるようになります。

CNNでは入力信号である生の画像データから始まり、各層ごとの畳み込みの処理が行われた後、たとえば最終的な出力として分類が行われる場合は、分類ラベルの情報にまで変換されていきます。その中でCNNの畳み込み処理の役割は、入力からある特徴を抽出し（この処理は**特徴量抽出**と呼ばれます）、抽出された特徴量を新たな画像形式の出力として、特徴量マップを生成することです。つまり、それぞれの畳み込み層では、特徴量マップ（第1層では入力データ）が入力となり、出力として新たな特徴量マップを出力します。層が増えるということは、それぞれの層の特徴が組み合わされていくことを意味するので、入力に近い層ではシンプルな特徴がとらえられ、層が重なるにつれて、より複雑な特徴をとらえることが可能となります。たとえば、画像に人間が写っていたとして、それが人であると識別させたい場合、入力に近い畳み込み層では、線や弧など非常にシンプルな特徴がとらえられ、層が増えていくにつれて、目や鼻などのパーツを抽出し、さらに深くなると顔全体など非常に複雑な組み合わせの特徴までを抽出できるようになっていきます。

今までの内容をより具体的にイメージできるように、ある論文で紹介されている各層の特徴量マップを可視化した例をもとに説明します。2013年にリリースされたこの論文のタイトルは「Visualizing and Understanding Convolutional Networks」です。

最初にこの画像処理で用いられているCNNの構造を簡単に紹介します。

今回説明に用いるCNNは、図4-19のような構造で構成されるZF netというものです。ここでは畳み込み層が階層化されることの意味を理解することが主題なので、ZF netの構造の詳細については省略します。図4-19では、Layer 1からLayer 5の各層で畳み込みやプーリングが行われ、各層の特徴量マップが生成されるということがわかれば十分です。

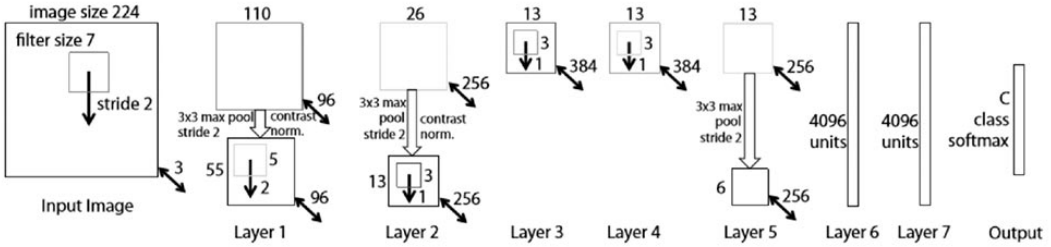


図4-19:ZF netの構成―[出典] Matthew D. Zeiler, Rob Fergus, "Visualizing and Understanding Convolutional Networks", Computer Vision - ECCV 2014, p.818-833

図4-20は、それぞれの層で畳み込みおよびプーリングの操作を行った結果、各層の特徴量マップがどのようなものであるかを可視化したものです。ここでは、各層の特徴量マップのうち、9つのトップの活性化パターンが表示されています。それぞれの層の表示結果を見るとわかるように、第1層の特徴量マップは、非常に基本的で線形的なパターンを学習しています。第2層では、円や矩形や縦縞のような第1層より少し複雑なパターンを学習し、第3層になると、ゼブラ模様や人の上半身や車のタイヤのような、物体のシンプルな形状のパーツのレベルを学習しています。第4層では、犬の顔や鳥の足などさらに複雑なパターンを学習し、第5層では、花びらや一輪車のようなかなり具体的でさらに詳細なパターンを学習しています。

以上のことから、CNNでは畳み込み層とプーリング層の組み合わせを積み重ねていくことで、複雑で細かいパターンを学習できるようになることが理解できます。

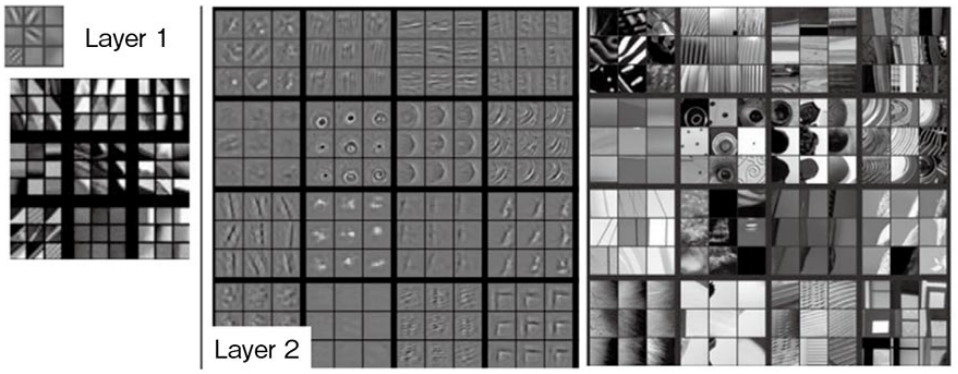


図4-20-1：ZF netを用いた1～2層の特徴量マップの例―［出典］Matthew D. Zeiler, Rob Fergus, "Visualizing and Understanding Convolutional Networks", Computer Vision - ECCV 2014, p.818-833

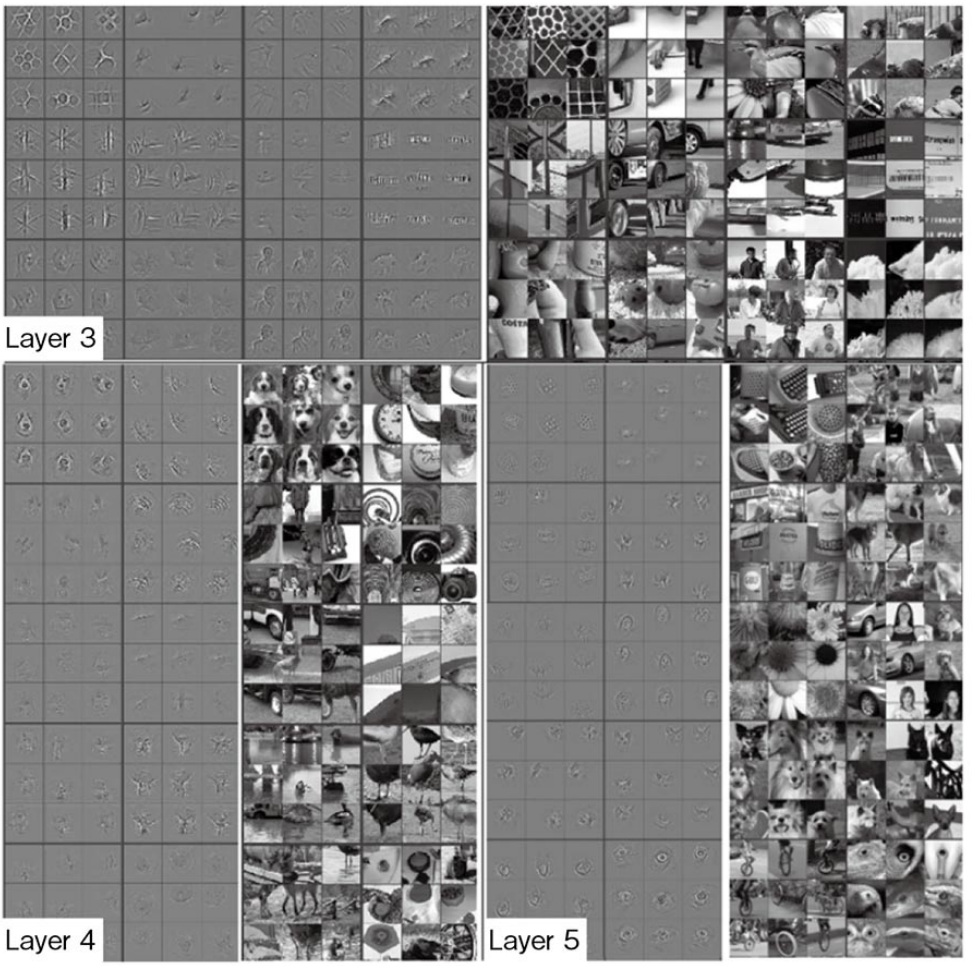


図4-20-2：ZF netを用いた3～5層の特徴量マップの例―［出典］Matthew D. Zeiler, Rob Fergus, "Visualizing and Understanding Convolutional Networks", Computer Vision - ECCV 2014, p.818-833

図4-20のような例からも、畳み込みの機能が画像の中のある特徴を抽出できることが視覚的に理解できるはずです。

▶4.3.4 パラメータ共有（parameter sharing）

パラメータ共有もCNNにとって重要な特徴の1つです。全結合型ニューラルネットワークでは、学習を行う際に、すべての入力データの1つずつに重みのパラメータが割り当てられるため、画像分類タスクには不向きであったことを思い出してください。全結合型ニューラルネットワークでは、たとえば入力画像における物体の位置が画像の中心から右上隅に移動すると、物体検出などの場合、位置の違いまで検出してしまう傾向があります。全結合型ニューラルネットワークとは対照的に、CNNでは、入力画像より小さい同じ重みの組み合わせであるカーネルを準備します。これで一度に入力画像の一部分だけを学習することが可能となります。画像全体の学習では、同一のカーネルをその画像上でずらしながら畳み込みの計算を実行します。全結合型ニューラルネットワークでは、それぞれのユニットの計算に用いる重みのパラメータは再利用されることなく一度きりのものでした。CNNでは、同じパラメータを別の部分でも再利用する、パラメータ共有という仕組みを用います。CNNでなぜパラメータ共有が重要かというと、もともとの入力画像のうちの一部分を学習できるので、画像の写っている位置に依存せず、特徴を学習できるようになるからです。また、パラメータ共有では、カーネルの使用により、パラメータ数を減らせるので、より高速に計算できるというメリットもあります。たとえば、1000画素の入力データに対して100個のノードを有する全結合型ニューラルネットワークで学習を行うには、重みのパラメータが1000×100の10万個になります。CNNの場合、1つのカーネルが3×3の9個、カーネル数が100個だった場合、900個のパラメータ数で特徴量の抽出が可能になり、パラメータ数を減らせるわけです。

パラメータ共有ではパラメータ数が減るので、別のメリットとして過剰適合（過学習）を抑えるという効果も期待できます。

パラメータ共有の仕組みのおかげで、学習の精度を向上させるために多量の画像のサンプル数を増やす必要がなく、さらには入力画像の大きさにも依存しないニューラルネットワークが実現できます。パラメータ共有はまた、過剰適合の抑制にも寄与するため、モデルの汎化機能を向上させることができます。パラメータ共有から得られる別のメリットとして、パラメータ数が抑えられるので、計算効率の向上とメモリ使用率の低下が可能になります。これらは、計算リソースに制限があるような業務アプリケーションでは、非常に重要な要素になります。

▶4.3.5 受容野

画像全体の中に含まれる局所的なパターンを認識する**受容野**という概念もCNNの重要な特徴の1つです。人の視覚の例で受容野を説明してみます。視覚のための網膜のさまざまな部位は神経細胞と結合されていますが、それらの神経細胞に影響を与える網膜上の範囲が受容野です。

CNNにおける受容野は、フィルタを通して出力の値に影響を与える入力層の領域と読み替えることができます。この感覚に留意しておくことで、畳み込み層などを多層にした場合の効果についての理解が深まるので、以降でその点をしっかり意識しておきましょう。

まず図4-21の例を見てください。

これまでの畳み込み処理の説明からわかるように、入力データが7×7、畳み込み層が単層、カーネルサイズが3×3、ストライドサイズ（ずらしの幅）が1となるCNNが存在する場合、出力となる特徴量マップは5×5となります。ここで特徴量マップのグレー領域に注目した場合、その受容野は、フィルタを介して見えている領域（つまりフィルタサイズ相当）です。そのため、入力データのグレー部分にあたる3×3の領域が、特徴量マップのグレー領域に対応する受容野になります。

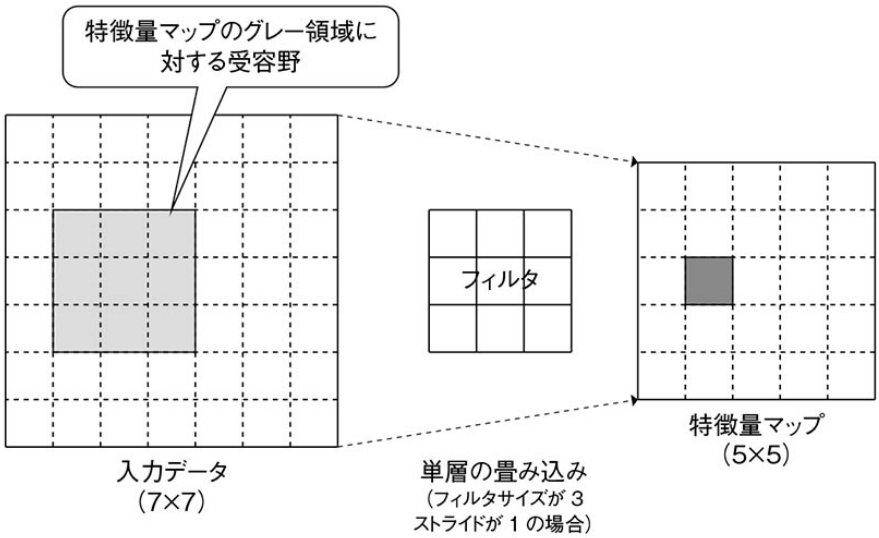


図4-21:CNNにおける受容野の説明

次に、図4-22のように入力データは10×10で、畳み込み層が3層あり、それぞれの層の畳み込み処理ではフィルタのサイズが3×3、ストライドのサイズが1であるとします。

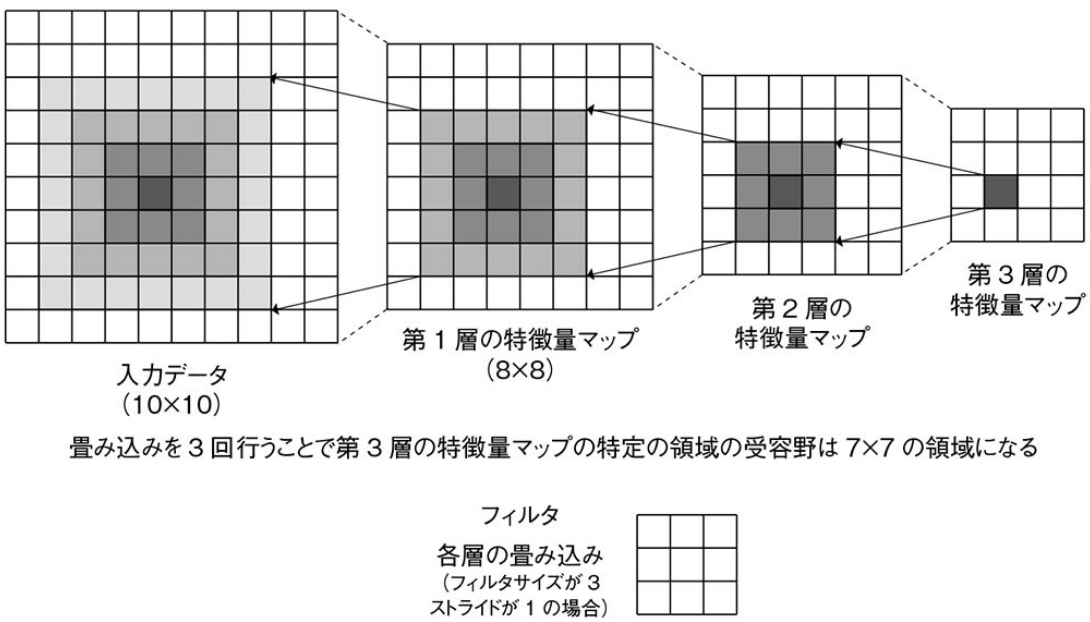


図4-22:複数の畳み込みによる入出力の間の受容野の領域拡大

3層の畳み込み処理が左から順番に行われると、図4-22のように第1層では8

×8、第2層では6×6、第3層では4×4の特徴量マップが出来上がります。ここでは受容野を説明するために、第3層の特徴量マップの濃色で示されているピクセルに注目してみます。この濃色部分に対応する第2層の受容野は、第2層の特徴量マップ（右から2番目の行列）のうち、グレーで示される3×3の領域です。同じように、第2層のグレー領域に対応する第1層（左から2番目の行列）の受容野は、5×5の領域です。さらに第1層のグレー領域に対応する入力データの部分は、一番薄いグレーで示される7×7の領域になっています。

つまり、第3層の濃色部分の特徴量の影響範囲は、3×3のフィルタの畳み込み層を介することで、入力データの7×7の受容野に増幅することができます。層を深くしていくことで、より広範な受容野を獲得できるわけです。

このことは、小さな畳み込みのフィルタ（3×3など）であっても、複数の畳み込み層を重ね合わせることによって、1つの層で大きなフィルタを使用するのと同等の効果を得ることができます。つまり図4-22の場合には、7×7のフィルタを単層で用いるのと同じような効果が得られるということです。

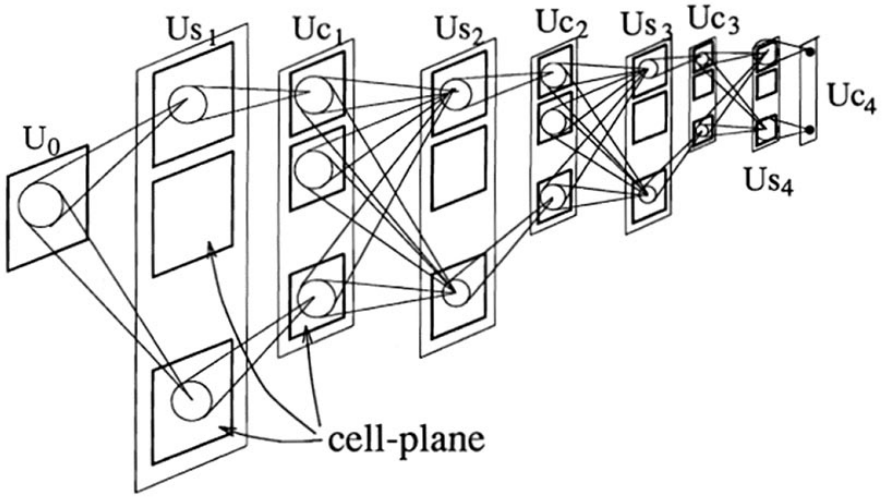
小さな畳み込みフィルタを使用することには、以下の2つの利点が存在します。

- 小さな畳み込みフィルタを複数重ね合わせることで、ネットワークが深くなり、複雑な特徴量の獲得が可能になります。
- パラメータ数を減らしながらネットワーク容量を増やせるようになります。

■ COLUMN　CNNの発想のもとになったネオコグニトロン

ネオコグニトロンは、1980年代に福島邦彦氏により提唱された、視覚パターン認識に関する階層型神経回路モデルで、CNNの発想のもとになっています。

下の図に示したように、ネオコグニトロンは、単純型細胞（S層）と、複雑型細胞（C層）を複数積み重ねた構造のモデルです。S層は畳み込み層に相当し、画像の特徴量を抽出する層です。C層はプーリング層に相当し、位置のずれを許容する層です。



図：ネオコグニトロンの概念図―［出典］佐藤俊治「視覚情報の統合過程に関する基礎的研究」(http://www.iic.ecei.tohoku.ac.jp/publication_data/225.pdf)

なお、この概念についてより詳しく知りたい方は、出典元の文献を参照してください。

図を見れば一目瞭然ですが、我々の視覚における認識が、CNNの発想のルーツになっています。

CNNでよく使われる機能の1つとして**パディング**機能があります。そこで本章の最後にその機能を解説します。

▶4.3.6 パディング

入力データのまま畳み込みを行うと、端の領域の値が他の領域と比べて畳み込み回数が少ない分、端の領域の情報が反映されにくくなります。パディングでは、端の領域の情報も反映されるように、図4-23のように周囲を0で埋めるような仕組みをとっています。

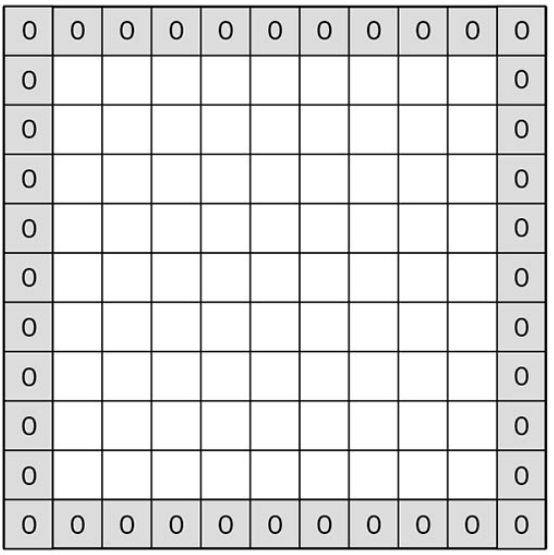


図4-23: パディングの説明

具体的に説明するために、次の図4-24のような入力データとフィルタの例で考えます。パディングなしで畳み込みを行うとして、図4-24の上の例のように、入力データの左上から右へ1ピクセルずつストライドする（ずらしていく）場合、左上の角のピクセルに対する畳み込み処理は一度きりになります。それに比べて、左端から2番目のピクセルは畳み込みの回数が2回、左端から3番目は畳み込みの回数が3回となり、角のピクセルよりも回数が増えます。そのため、特徴量として利用される頻度は、角のピクセルが他よりも少ないという問題が生じます。それに対して、元の入力データの周囲を0で埋めることにより、図4-24のように元の角のピクセルであっても合計4回の計算が行われるので、元の角のピクセルの特徴がより反映されるようになります。

図4-24の例は0パディングと呼ばれ、画像の周囲を0で埋めています。

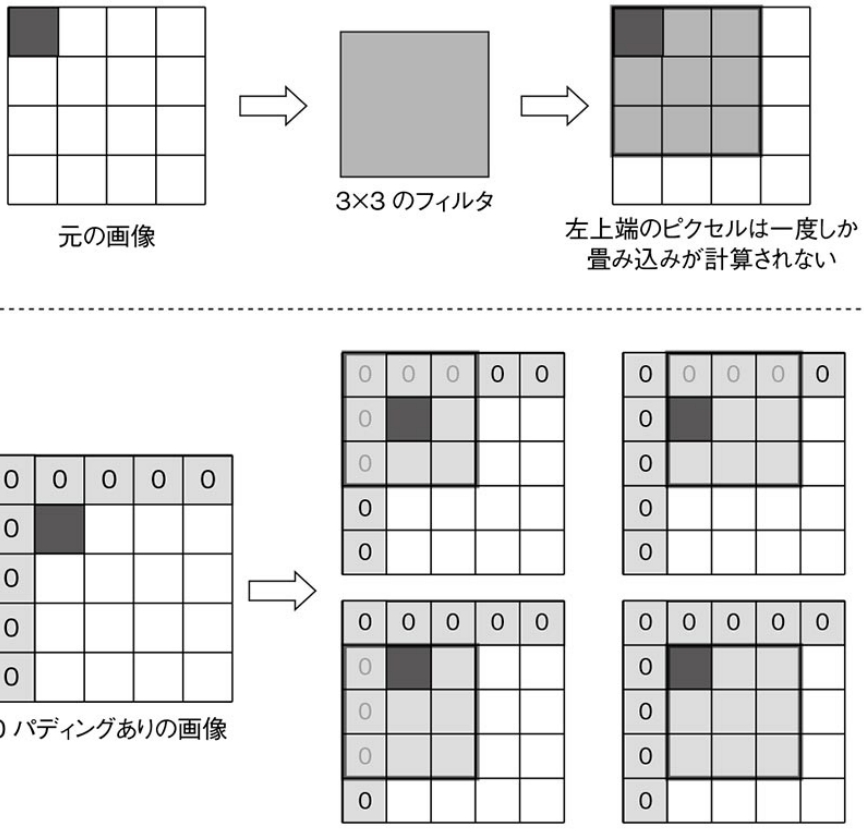


図4-24:画像の周囲を0で埋める例（下）と埋めない例（上）

この図のようにパディングされた領域からフィルタの畳み込みを実行していくことにより、入力データの端の値も、ほかの領域と同程度の回数で畳み込みが行われるようになり、端の特徴も反映されるようになります。

すでに読者の中で気づかれた方がいらっしゃると思いますが、0パディングの場合、端のピクセル値がたとえば255だったとすると、画像の周囲に0パディングを配置することにより、元の入力データにはなかった周囲の0と画像の端の255の値の差によるエッジの特徴をとらえてしまいます。そのため、0パディングのほかに、平均値パディングなどの方法があります。ただし、平均値パディングには平均値を出すという計算コストがかかるため、性能とのトレードオフがあることに注意してください。また、図4-24の例では、周囲のパディングは1ピクセルだけでしたが、複数ピクセル分をパディングすることもあります。適切なパディングは、使用するフィ

ルタのサイズや画像の特性によって異なるので、用途に合わせて選択する必要があります。

CNNに関する説明は以上です。なお最後に、CNNの算術的な詳細情報は、さまざまな書籍やインターネットから入手可能なので、必要な方はそちらを参照してください。また、英語の情報ではありますが、本節の内容に関して、非常に優れた可視化を行っている下記URLのブログがあります。本章の説明の補足としてぜひ参照してください。

「A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way」
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

4.4 まとめ

本章では、畳み込み層、プーリング層を中心にそれぞれの役割と畳み込みの操作、さらにその特徴について見てきました。畳み込みは画像などの入力データの特徴を抽出するのに適した仕組みであり、プーリング層は位置ずれなど物体検知に不要な情報の学習を抑制する仕組みであることが理解できたと思います。実際にCNNを利用した際、各層のハイパーパラメータのチューニングなどでうまくいかないといった場合に、本章で説明した仕組みを理解しておくことで解決の糸口を見つけることができれば幸いです。

MEMO

第5章

再帰型ニューラルネットワークのメカニズムと意味をとらえる

適切な利用のための直感的理解

Deep Learning Project Template

本章は、前章のCNNと同じくらい有名でデータ分析の場で広く利用されている**再帰型ニューラルネットワーク**（Recurrent Neural Network：以降、**RNN**）について説明します。

5.1 本章の目的

RNNは、**連続データ**（以降、**シーケンシャルデータ**）からパターンを学習するために特化したタイプのニューラルネットワークです。CNNが画像認識などの視覚的なタスクに用いられる一方、RNNは、シーケンシャルデータの処理を実行することが目的です。本章では、シーケンシャルデータがどのようなものか、またそのようなデータを処理するためにRNNのような新たなアーキテクチャが必要とされる理由を説明していきます。本章は、データ分析を行う際に、どのようなシーンでRNNを用いるべきかを正しく理解していただくことを目的としています。

最初に、RNNの一般的な概念の説明を行います。その後でRNNの詳細なメカニズムを説明するために、RNNの数多くある種類のうち、**LSTM**（Long Short-Term Memory）ニューラルネットワークに注目して説明します。LSTMを用いる理由は、RNNの中で非常に有名なアーキテクチャであり、また何十億個のデバイスを日々使うような、優れたエンタープライズのアプリケーションで利用される基盤になっているためです。LSTMは実際、すでに幅広く用いられていて、機械翻訳、音声認識、文字認識など、企業やコンシューマ向けのアプリケーションで最も一

般的に使用されているニューラルネットワークのアーキテクチャの1つです※1。

ここでは、LSTMについてどこかで聞いたことがあるか、そのメカニズムについて書かれた書籍などを読んだことがあることを前提に話を進めていきます。ただし、その概念や仕組みを完全に理解している必要はありません。実際、本章では、LSTMを直感的に理解するのが難しい方々に、より深い洞察を提供することを目的としています。

LSTMについて書かれた学習用の情報の多くは、LSTMのシンプルな機能操作についての説明にとどまっていて、LSTMが果たしている役割についての解釈に触れられていません。または、多くの場合、「データ」がネットワーク操作全体で流れる順序についてはカバーしていますが、データ自体の内容については説明されていません。結果的に、その概念自体にあまり慣れていない方にとっては、すぐにLSTMの設定方法について明確なイメージを描くのが難しく、特にネットワークを流れるデータの次元についての感覚を身につけるのは難しいかもしれません。理論と実践の間にはまだギャップが残っています。次のステップに従ってLSTMの果たす役割まで学ぶことによりギャップを埋めていくことで、これらの困難を取り除いていきます。

本章は以下のように構成されています。

※1 ［参考文献］Juergen Schmidhuber「Our impact on the world's most valuable public companies (Google, Apple, Microsoft, Amazon, etc)」http://people.idsia.ch/~juergen/impact-on-most-valuable-companies.html

それではさっそくRNNの内容から見ていきましょう。

5.2 シーケンシャルデータとRNNの概要

RNNで用いられるシーケンシャルデータとは、その名のとおり連続するデータを意味し、前後の関連性が存在するデータのことです。代表的なものに、時系列データや文章データなどがあります。RNNを使って正しくモデル化が可能な分野の例は多数存在します※2。最初に思い浮かぶのは、時系列データを使った一般的なシステムのログ分析や株価予測などの時系列分析の分野です。またもう1つよく知られている分野は、自然言語分析です。RNNは、機械翻訳などの自然言語処理（NLP）のタスクによく使用されます。NLPでは、文は、前後の単語の関係性を示すデータであるためシーケンシャルデータとして表現できます。これからシーケンシャルデータとRNNについて説明していきます。

▶5.2.1 シーケンシャルデータ

一般的に、シーケンシャルデータは値の配列で表現され、値は順列の性質を持ち、それぞれの値には時間（タイムスタンプ）のインデックスを付けることができま

※2 ［参考文献］Juergen Schmidhuber「Our impact on the world's most valuable public companies (Google, Apple, Microsoft, Amazon, etc)」http://people.idsia.ch/~juergen/impact-on-most-valuable-companies.html

す。たとえば、[x_0, x_1, x_2, …, x_t]です。多くの場合、値x_tの現在の時間tは、前の値(x_0, …, x_{t-1})とある程度の相関関係があります。つまり、シーケンスの順序が重要となるようなデータ構造をしています。NLPの例では、文は単語のシーケンス表現としてイメージすることができます。

この概念を具体的に説明するために“Alex has been living in Japan for 20 years and he speaks fluent Japanese.”という例文を取り上げます。例文は、[x_0=“*Alex*”, x_1=“*has*”, x_2=“*been*”,… , x_{13}=“*Japanese*”, x_{14}=“.”]のように連続したものとして表現できます。再帰型ニューラルネットワークは、データの時系列の間に相関があるようなタイプのデータをモデル化するために発明されました。視覚的に理解するための下の図5-1を参照してください。

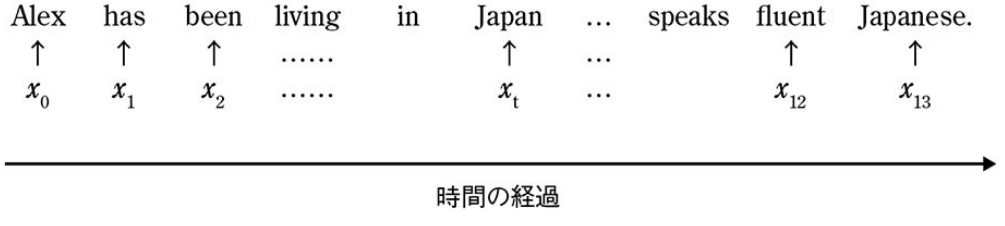


図5-1:シーケンシャルデータの例

▶5.2.2 RNNが必要な理由

なぜ全結合型ニューラルネットワークやCNNとは別の新たなニューラルネットワークが必要なのか疑問に思われる方もいらっしゃるでしょう。ここでは具体的にRNNのイメージを持っていただきやすいように、機械翻訳の例を用いて解説していきます。

英語から日本語への機械翻訳エンジンを構築するため、標準的な全結合型ニューラルネットワークを用いて、このタイプのシーケンシャルなデータをモデル化する必要があると仮定します。我々が最初に直面する問題は、全結合型ニューラルネットワークでは、入力のデータサイズと出力のデータサイズを事前に固定する

必要があり、異なる長さのデータをモデル化するのが難しいということです。もちろん、テキストのような文章のデータではデータの長さは固定できないため、常に入力と同じ長さの文を期待することは現実的ではありません。2番目の問題は、例文の最後の“Japanese”という単語の訳として、国籍としての「日本」を指すのではなく、言語としての「日本語」を指すものである理由を正しく区別するのが難しいことにあります。人は、その単語より前の情報（単語）から、“Japanese”が言語としての「日本語」を指していることを理解します。全結合型ニューラルネットワークでは、すべての入力が同時に与えられるため、前の入力で示された動作（予測）を学習することが非常に難しくなります。自然言語処理で直面する別の問題は、入力を理解し、それらを出力文に翻訳するために、モデルが言語の文法を学習する必要があるということです。全結合型ニューラルネットワークを用いて、単語の前後文脈をとらえ、適切な品詞などの文法の学習を実装する方法は容易に思いつかないでしょう。つまり連続するデータであるシーケンシャルデータの性質を学習していくのに、全結合型ニューラルネットワークは適していません。そこでシーケンシャルデータの学習に適したニューラルネットワークが必要ということになり、RNNというアーキテクチャが考案されました。

▶5.2.3 RNN アーキテクチャの概要

これまでの説明で、このようなシーケンシャルなモデリングの問題を解決するための新たなアーキテクチャが必要であることを理解できたので、本項からは、この問題を解決するために考案された再帰型ニューラルネットワークの一般的なアーキテクチャについて見ていきます。

最初に、再帰の大まかなアイデアを把握することに注力するため、データが数値形式でどのように表示されるか、それぞれの操作が代数的にどのように構成されるか、などの詳細については一度忘れてください。次の図5-2は、私たちが最終的に理解すべきRNNの構成図となります。

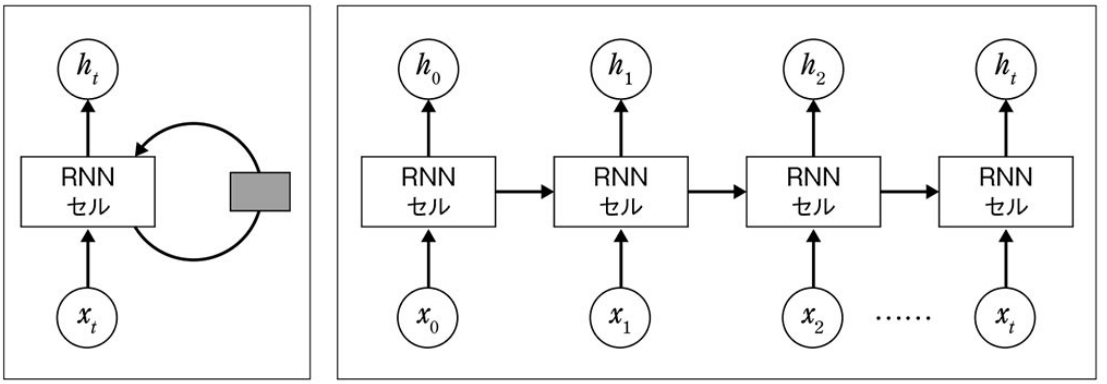


図5-2:RNNの構成図

多くの場合、RNNはまったく同じ機能を示す上の左右2つの図のいずれかの方法で描かれます。上図の右側の図は、多くの時間ステップの操作が図示されており、シーケンシャルな操作の多くが線画で示されたRNNの各セルを示したものになります。図の四角のボックスで示されるRNNセルは、RNN固有の操作が行われるコンテナのようなものです。左図の形式はRNNセルにループの矢印を記述することにより、右図のような繰り返しを省略しています。

図のx_xが入力データ、RNNセルがRNNのメインブロックで、ニューラルネットワークの重みと活性化関数が含まれます。h_xは出力を示します。左図のループを構成している矢印、右図のRNNセル間で接続される矢印は、1つの時間ステップから次の時間ステップへ情報が伝搬することを示しています。

展開されたRNNの操作の図を用いることで、学習アルゴリズムの再帰の性質をイメージするのには役立ちますが、それぞれのRNNセルで用いられる重みに関しては、展開されたRNNの図を見ると、それぞれのRNNセルで異なる訓練可能な重みが設定されていると勘違いしがちです。RNNではすべてのRNNセルで重みを共有します。CNNのカーネルの重みは入力画像全体で共有されていました（つまり、入力画像のすべてのピクセルに対して重みがそれぞれで割り当てられずに共有されることを意味します）。これと同様にすべてのRNNセルで同じ重みの組み合わせが用いられます。したがって1番目のRNNセルの重みは、2番目または一般化したt番目のRNNセルの重みと同じものになるのでその点に注意して

ください。

▶5.2.4 RNNにおけるデータフロー

さっそく、RNNにおける入出力のデータのフローから説明しましょう。

RNNのニューラルネットワークは、連続するいわゆるシーケンシャルデータをモデル化するのに使用されることを思い出してください。RNNでこのようなシーケンシャルデータを処理するためには、シーケンスが続く限り、入力データのひとかたまり（**チャンク**）を1つずつRNNのセルに与える必要があります。処理の途中で、それぞれのセルは前のセルから過去の入力情報を取得し、現在の入力情報を追加して、それ自身も再び次の時間ステップに渡される出力を生成する、といった仕組みが実装されています。

たとえば、前の例文を使って、今度は、品詞（名詞、動詞、副詞など）のタグ付けをするシステムを構築するとします。私たちはすでに文は単語が連続することで形成されることを見てきました。たとえば、[x_0= “$Alex$”, x_1= “has”, x_2= “$been$”,… , x_{13}= “$Japanese$”, x_{14}= “.”]です。x_0=“$Alex$” からx_{14}=“.”で始まる単語ごとにそれまでの単語の処理結果と合わせて処理することで再帰が起こり、その間、それぞれのセルは、連続する関係性についての処理を行うために、前の単語の処理結果と現在の単語の情報を取得します。この様子を以下の図5-3で示しています。図の左端では、入力x_0に文章の冒頭の単語にあたる“Alex”が入力され、x_1には2番目の単語にあたる“has”が入力され、というように、文章の各単語が順番に各RNNセルに入力されます。また、左から2番目以降のRNNセルには、前段のセルから矢印で示される入力も存在し、これにより前段の入力情報も伝搬してきます。図の左から2番目のRNNセルには“Alex”の情報が伝搬し、3番目のRNNセルには、”Alex”と”has”という2つの情報が入力として伝搬してきます。それぞれRNNセルからの出力は、特定の入力後の品詞のカテゴリに対応していて、1番目のRNNセルからの出力は“名詞”、2番目のRNNセルからの出力は“動詞”という情報が出力されます。この例は、品詞を出力するような例で

すが、RNNのアーキテクチャは、機械翻訳、音声認識など解決しようとする課題によって異なるため、バリエーションが存在することに注意してください。5.4節でその他のバリエーションについて簡単に説明します。

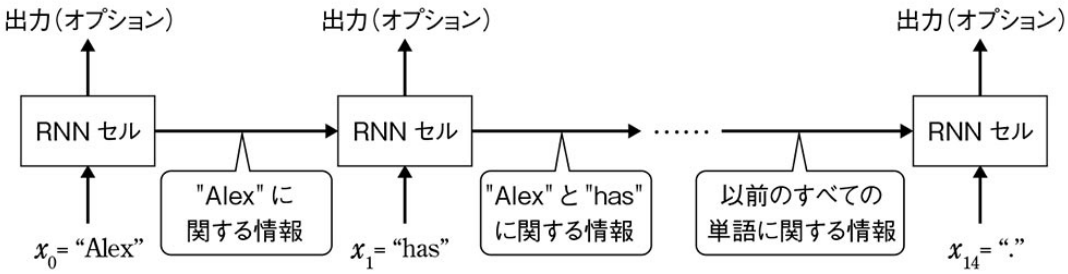


図5-3:RNNを用いた品詞特定のフロー例

▶5.2.5 RNNのメモリ機能

データの前後の関係性を理解するためには、現在のRNNセルに前段のRNNセルの出力結果を再帰的に用いる操作が必要です。RNNでは、RNNの計算結果を再帰的に用いるために、出力を記憶しておくメモリの機能を有しているという点が重要になります。機械翻訳の課題について説明した5.2.2項の説明を思い出してください。"Japanese"という言葉を「日本語」と、国籍としての「日本」のどちらで訳するかを伝えるには、"Alex has been living in Japan for 20 years"という文の文脈を理解するためのスキームが必要であることを学びました。RNNでは、国籍としての「日本」の意味を正しく判断するために、"Japanese"という単体の単語だけではなく、その前の単語との関係性を記憶し学習していくのでした。

文脈を的確にとらえ正しく目的となる単語の意味を高い精度で予測するためには、現在の単語と前の単語の関係性について、過去の時間ステップのうち重要な情報を抽出する方法を学習し、抽出された情報をメモリに記憶していく必要があります。LSTMを含むRNNの数多くの種類のバリエーションでは、データのシーケンシャルな関連性に関する特徴を抽出するために、RNNセル内で複雑な計算が行われますが、本質的な部分はかなりシンプルなものです。現在の単語の入力

データとメモリに記憶されている過去の情報を入力として使用し、RNNセルでの計算結果の情報を利用できるようにするために、出力結果をメモリに蓄積していくというものです。つまり、RNNでは、シーケンシャルデータの前後の関係性をとらえるために現在の入力データのみならず、RNNセルの計算結果についてもメモリに蓄積する機能を有し、再帰的に利用していく仕組みを特徴としたものなので、メモリの機能が重要になります。

■ COLUMN　word embedding

第3章で紹介した、カテゴリデータを数値演算が可能な形に変換する仕組みであるone-hotエンコーディングを、単語を表現するケースに当てはめてみると、たとえば1万語の単語があるとしたら、1万次元のベクトルとして各要素をそれぞれの単語に割り当てることになります。

この方式では、同一単語かどうかの比較しかできず、さらに、未知の単語や新しい単語を追加しようとするとベクトルの次元が増えてしまうため、データ圧縮効率の悪い実装になります。

そこで、もっと単語の特徴をとらえやすいベクトル表現に変換して、演算処理を可能にする**word embedding**という方式が考案されました。

word embeddingとは、単語を一定数（一般的に200から1000次元くらいの大きさ）のベクトルに変換する手法です。one-hotエンコーディングでは、1万語を表現するのに単語数と同じだけの1万次元が必要でしたが、word embeddingを用いることで1000次元程度に表現を圧縮できます。

word embeddingにはさまざまな種類がありますが、ここでは理解がしやすい**Word2Vec**という方式を説明します。

Word2Vecは、意味が似た単語の周りには同じような単語が出現するようなベクトル表現を実現する手法です。

このような表現にすると、単語同士の意味の近さを比較したり、足し引きなどの演算ができるようになります。

具体的なイメージとして、"王子様"－"男"＋"女"＝"お姫様"というような単語の意味に基づき単語同士の関係性を調べられるようになります。

以上のことから、自然言語処理ではword embeddingの手法を用いることで、機械学習なども含む数値演算で単語を扱いやすい形式に変換でき、さらにデータの次元数も減らせるため計算効率が飛躍的に良くなります。

▶5.2.6 RNNの操作フロー

RNNセルの構造とそれぞれのコンポーネントの機能は、次の図5-4に示すようにとてもシンプルです。RNNセルは、「入力データx_tと前段の出力結果（隠れ層の出力）h_{t-1}を足し合わせる機能」と「tanhの活性化関数で勾配計算を行う機能」で構成されます。その計算結果が、次のRNNセルの入力として再帰的に利用されます。図は出力が利用される例を示していますが、RNNセルの出力結果のうち1つは上の○で示されるh_tとして出力されます。それぞれの時間ステップでの出力が必要か否かは解くべき課題によって異なります。RNNのバリエーションについては本章の最後で説明します。

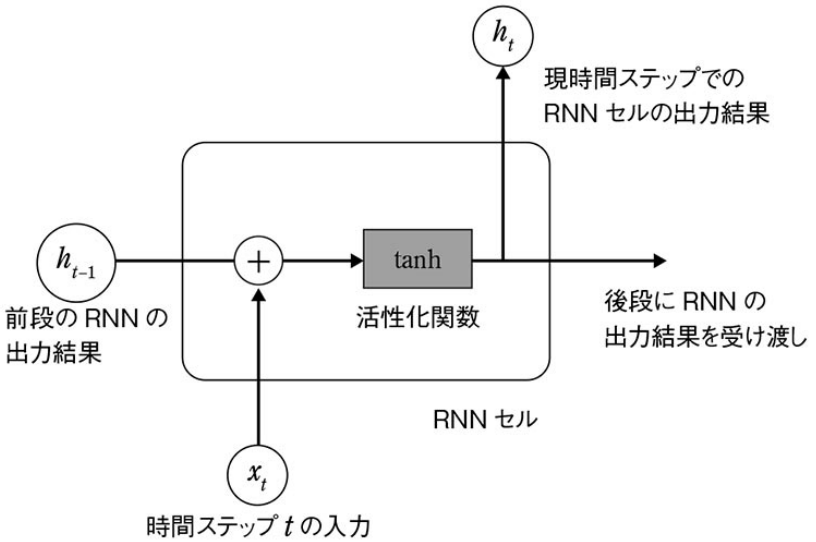


図5-4：RNNセルのコンポーネント

RNNセルでは、上図のとおり活性化関数に**ハイパボリックタンジェント**（tanh）関数が使用されます。tanhは次の図5-5のようにすべての入力値を−1から1の範囲内の値に変換するため、RNNセルの出力は必ず−1から1の値となります。たとえば、RNNセルを1時間ステップ通るたびに0.9の値が返ってくるような計算結果になったとすると、10時間ステップ通った場合は単純計算で約0.349（0.9の10乗）の値になるため、最初の時間ステップの情報が10時間ステップでは約35%しか伝搬しないことを意味します。これが100時間ステップだと約0.0000266（0.9の100乗）となり、ほとんど情報が伝搬しないことがわかります。

学習の際の誤差逆伝播でも事態は変わらず、RNNセルの場合、100時間ステップなどでは学習が進まなくなります。このように勾配がどんどん小さくなり消失していくために学習が進まなくなることは**勾配消失問題**と呼ばれます。シンプルなRNNでは勾配消失問題が発生するため、大規模なネットワーク構成をとることができず、用途がある程度限定的になりますが、それを解消するために次節で紹介するLSTMが登場します。

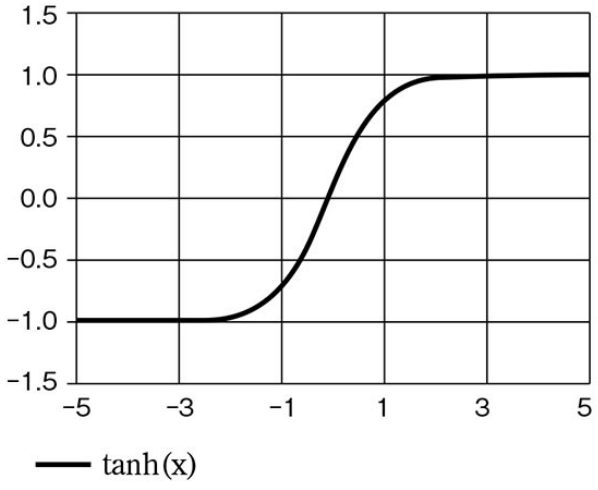


図5-5:tanh関数のグラフ

RNNの概要についての説明は以上となります。シーケンシャルデータとはデータの前後の関係性を有するもので、その関係性を正しく抽出するために、前後の関係性をメモリに蓄積する機能を有するRNNが必要になったということを本節で説明しました。

5.3 LSTM

これから、勾配消失問題を解消するために開発されたLSTMネットワークに焦点を当てて解説していきます(それに伴い今までRNNセルと呼んでいた部分は**LSTMセル**に置き換えます)。まずLSTMがRNNと異なる部分の概要について解説した後、LSTMセル内のそれぞれのコンポーネントの操作を説明していくことで、LSTMの仕組みの理解を深めていきましょう。

▶5.3.1 LSTMの全体構成

本節では、RNNアーキテクチャの一種であるLSTMの全体構成について説明します。

前節の説明から、先のRNNセルでも数十時間ステップくらいの短い時間ステッ

プの場合には、勾配消失の影響が小さいために対応できますが、100時間ステップなどでは対応ができないことがわかりました。

LSTMとは、Long Short-Term Memoryの略です。RNNのバリエーションの一種で、RNNで生じる勾配消失問題を解消するために考案されたニューラルネットアーキテクチャになっています。RNNセルに換わるLSTMセルの各コンポートの役割を説明していきます。

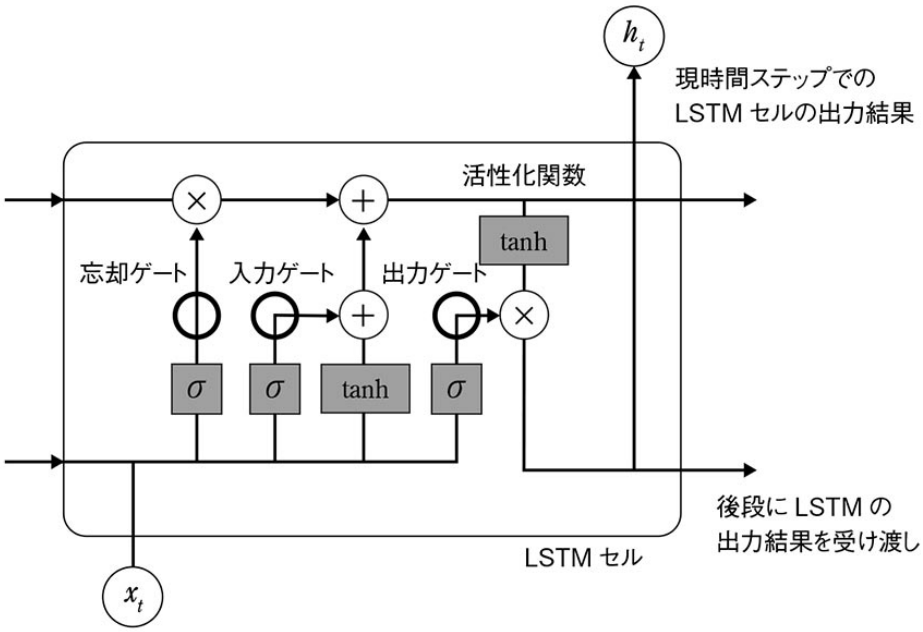
▶5.3.2 LSTMセルの機能と動作

これから、LSTM内のそれぞれのコンポーネントの機能についての詳細な動作を順番に解説します。できるだけ直感的に理解できるように説明していきます。LSTMセル機能の概要の説明に集中するために、ここではデータの次元についての説明は省略します。ただし、実践的な例や、LSTMのその他の実装の段階になってくると、セル内のそれぞれのデータ構造の次元を理解しようとする場合に問題が生じる可能性が非常に高くなります。そのため、データの次元については次節で説明します。したがって、今のところは次元についてはあまり気にせず理解を進めてください。参考までにこの節の当該部分の内容については、主にChris Olah（クリス・オラー）氏による有名なブログ投稿から着想を得ています。ご興味がある方はそちらもご参照いただけると幸いです※3。

LSTMは、RNNの概要で説明したRNNセルの部分が、以下の機能で構成されるLSTMセルに置き換わったものです。基本的なRNNの機能である再帰的なデータ入力を有すること、さらにそれらをメモリで蓄積する仕組みであることに変わりはありません。

※3 Chris Olah「Understanding LSTM Networks」https://colah.github.io/posts/2015-08-Understanding-LSTMs/

図5-6:LSTMの概略



$$
\begin{aligned}
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
o_t &= \sigma(W_o [h_{t-1}, x_t] + b_o) \\
\tilde{c}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{c}_t \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

図5-7:LSTMセルの機能

まず図のゲートの役割として用いられる図のσについて説明します。これは**シグモイド関数**で、第3章の活性化関数で説明したとおり、次の図5-8のように、入力値を0から1までの数値に変換し出力します。シグモイド関数は各コンポーネントをどの程度通すべきかを表現するために用いることができるため、ゲートの役割を果たします。具体的には0は「何も通さない」としつつ、1は「すべてを通す」というように利用します。

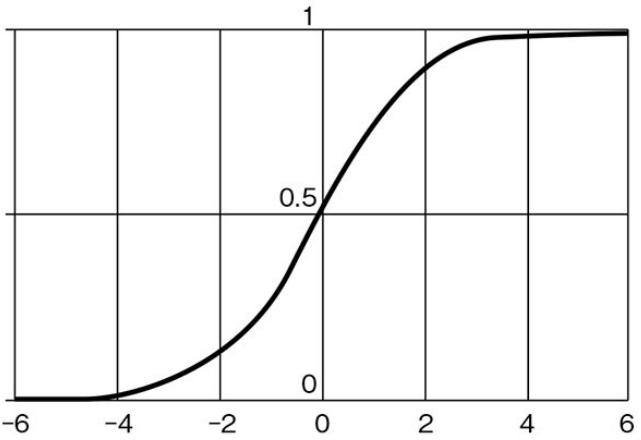


図5-8:シグモイド関数

単一のLSTMセルの操作は、以下に示される4つのステップで記述できます。

- **[第1ステップ:忘却ゲート]** 古いメモリから破棄すべき情報を決定する機能
- **[第2ステップ:入力ゲート]** 入力から新たなメモリに追加すべき新たな情報を決定する機能
- **[第3ステップ:メモリセル]** LSTMセルの状態を更新する機能
- **[第4ステップ:出力ゲート]** 出力すべきものを決定する機能

わずか4つのステップであり、それぞれの機能の説明は難しくありません。そのため、できるだけ直感的な説明となるよう努めます。それでは、それぞれのステップの詳細な説明を見ていきましょう。

▶5.3.2.1 第 1 ステップ：忘却ゲート（Forget Gate）

入力シーケンシャルの時間ステップごとに、すべての情報を均等に記憶していくことは必ずしも有効ではありません。ネットワークは、特定の時間ステップでさらに注力を払う必要がある情報に集中できるように、すでに不要となっている情報は忘れるようにする必要があります。まさにこの目的のために、忘却ゲートの操作

が実行されます。このネットワークは、適切なタイミングで情報を忘れることを自ら学習します。忘却ゲートの出力（値は0～1の間で、0は忘却、1は維持を意味する）は直前のセルのステート（図5-9のC_{t-1}）と直接乗算され、何を保持し、何を破棄するかを選択します。この概念を説明するのに適した例は、自然言語処理で文の主題が変わる場合です。前のテーマを忘れて、新たに提示されるテーマにもっと注意を払うようにしたいからです。ここでの入力、出力、操作について端的に表現すると以下のようになります。

●入力:

x_t:（1つの単語のような）現時点の時間ステップの入力

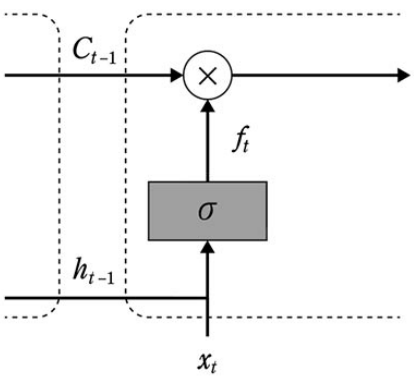
h_{t-1}:前の隠れステート（hidden state）

●出力:

f_t:0～1の値。何を忘れて何を維持するかを決定。これは第3ステップで使用される

●操作:

このゲートには、訓練可能な重みとバイアスがあり、使用される活性化関数はシグモイド関数で、数値を0～1の間の値に圧縮します。図5-9のダイアグラムでは、LSTMのセル全体のうち、忘却ゲートの部分のみが示されています。h_{t-1}とx_tの入力が連結され、重みが乗算されます。シグモイド関数の活性化の前にバイアスが追加されています。これはニューロン内の標準的なフィードフォワードニューラルネットワークの操作と非常によく似ていることに注意してください。



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

図5-9:忘却ゲート関連のコンポーネント

▶5.3.2.2　第２ステップ：入力ゲート

このステップでは、ネットワークが入力から新たなメモリに追加するべき新たな情報を決定します。ここでは次の2つの別々の操作が発生します。①入力をアクティベーションすること。これは、生データを適した形式に変換し、学習させて出力を得ることを意味します。②入力ゲートの操作では、入力からどの情報をセルのステートに追加／選択するかについて学習します（値は0～1の間です。つまり0は追加されず、1は追加されることを意味します）。その後、入力のアクティベーションからの情報のみがセルのステートに伝達されるように、第3ステップでアクティベーションと入力ゲートからの2つの出力を乗算します。第1ステップと同じ例を使って、入力文に新しいテーマが現れたときに、この操作手順で新たなテーマに関する情報を取得してから、のちの第3ステップで新たなメモリにこの情報が追加されます。

●入力:

x_t:（1つの単語のような）現時点の時間ステップの入力

h_{t-1}:前の隠れステート（hidden state）

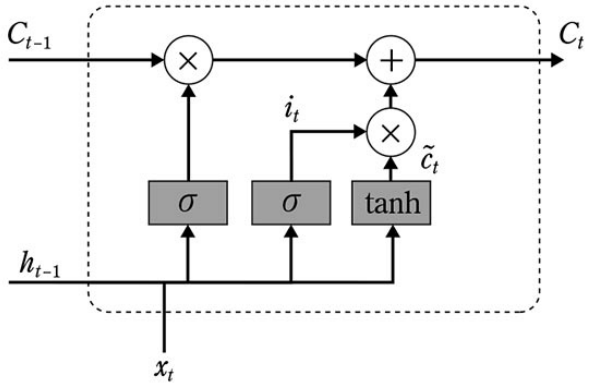
●出力:

i_t:入力ゲートの出力。値は0～1の間。新たなメモリに何を追加するかを決定する

$\tilde{c}_t$:入力の活性化値

●操作:

入力ゲートと活性化の操作の両方で、独自の訓練可能な重みとバイアスが存在します。忘却ゲートと同様に、入力ゲートの出力値は、シグモイドの活性化関数を用いて0～1の値になります。入力の活性化関数としては、tanh関数が使用され、－1から1の間の値に圧縮されます。LSTMセルの操作の一部と、このステップで使用される式については、図5-10を参照してください。



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

図5-10:入力ゲート関連のコンポーネント

▶5.3.2.3 第3ステップ:メモリセル

このステップは、前述の2つのステップを組み合わせたものです。「ネットワークが1つ前のセルの状態である第1ステップの情報を破棄するか否か」と「第2ステップの入力から追加する新たな情報」を決定した後、これらの2つの情報を組み合わせて新たなセルの状態(sell stateと呼ばれます)を生成しメモリに記憶します。このセルの状態は、乗算と加算の単純な線形変換のアップデート操作によってのみ更新されます。

●入力:

f_t :第1ステップからの情報

C_{t-1}:直前の時間ステップからの情報

i_t:第2ステップからの情報

$\tilde{c}_t$:第2ステップからの情報

●出力:

C_t:新たにアップデートされたメモリ(セルのステート)

●操作:

この操作については、次の簡単な式が最もよく表しています。

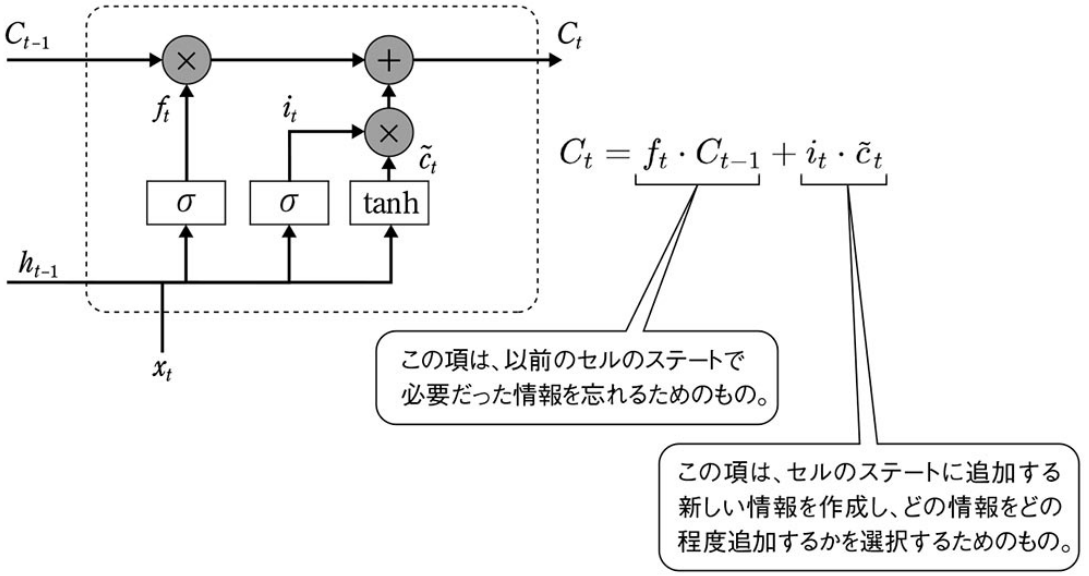


図5-11:LSTMセルの状態更新

▶5.3.2.4 第 4 ステップ：出力ゲート

最後に、出力ゲートの操作は、出力ステート（出力状態）と、隠れステート（隠れ状態）としてLSTMのセルが何の情報を出力するのかを決定します。これらの情報は次の図5-12に示す（両方ともh_tの）値と同じです。多くの場合、出力ステートは予測を行うために（ソフトマックス層などの）何らかのタイプの出力層に渡され、隠れステートは次の時間ステップのLSTMセルに渡されます。

出力値のh_tはセルの状態C_tに基づいていますが、出力される前にいくつかの修正がかかります。最初は、**要素ごと（pointwise）**のtanh操作によって−1から1の値に圧縮されます。そして、さらに出力ゲートの操作の直接的な結果のo_tが乗算されます。他のゲートと同じように、出力ゲートは入力としてh_{t-1}とx_tを取得して、（出力するものを選択するために）シグモイド演算によって0～1の値を出力します。また、訓練可能な重みやバイアスも存在します。

この圧縮操作は要素ごと（pointwise）の操作であることに注意してください。つまり（入力のアクティベーションのステップである）第2ステップで発生したtanh演算とは対照的に、この圧縮操作には訓練可能な重みが存在しません。つまり、

ネットワーク全体で常に同じ圧縮操作となります。

●入力:

x_t:(1つの単語のような)現時点の時間ステップの入力

h_{t-1}:前の隠れステート(hidden state)

●出力:

h_t:(出力ステートでもある)隠れステート

●操作:

次の図と式を参照してください。

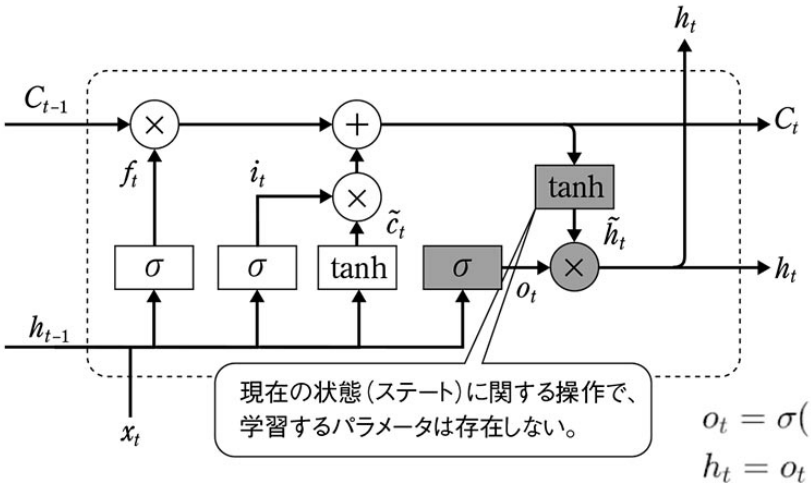


図5-12:出力ゲート関連のコンポーネント

▶5.3.3 データの次元

LSTMセル内の各ステップのフローに従ってLSTMの機能を説明しましたが、LSTMのネットワークを実装したり使おうとしたりする場合には、それでも適切な方法が見い出せない可能性があります。このような状況に陥る理由は、多くの場合、LSTMのセルで用いられるデータの次元を理解することが難しいからです。そこで、入力データに始まり中間層にあたる各時間ステップの状態のデータ、さら

には出力データがどのように見えるのかについて、LSTMの操作を通じてデータの形状を追跡し可視化していくことで、説明していきます。追跡と可視化にはさまざまな実現方法がありますが、読者の方々がすでに学ばれた4つのステップをベースに説明します。各ステップの詳細な説明に入る前に、まずは各ステップでの操作で共通する入力データのベクトル表現および隠れ層のユニットのサイズについて理解しましょう。

▶5.3.3.1 入力データのベクトル表現

RNNでは、入力x_tはベクトル形式で表現されます。ここでは、入力データがどのようなベクトル形式であるのかについて、自然言語処理（NLP）の例をもとに説明します。NLPの多くでは、特定の時間tの処理について考える場合、その時間ステップのLSTMセルへの入力は、1つの単語が、入力データとして用いられます。この入力データをすべての時間ステップで俯瞰したものがシーケンシャルデータである、という解釈も可能です。各時間ステップのLSTMセルへ各単語がシーケンスデータとして入力されることで、たとえば1つの文章が形成されます。大まかなイメージとしてはこれでよいのですが、実際の入力データは、先のコラムで説明したとおり、辞書の中の単語を参照するために、整数のインデックスを使うという単純なものではなく、単語の分散表現を意味するword embeddingを使って、各単語をベクトルとして表現することがよくあります。word embeddingで単語を表現することにより、単語は埋め込み空間（embedding space）によって豊かな文脈に相当する意味を持つことができ、次元数も減らせるため、よく用いられる手法となっています。その詳細については、下記リンクの情報を参照してください。

「**Understanding LSTM Networks**」（**Chris Olah**）

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

▶5.3.3.2 隠れ層のユニットのサイズ

次に隠れユニットのサイズについて説明します。隠れユニットのサイズは、ディープラーニングのフレームワークにおけるLSTM層のパラメータ数を意味するもの、と考えると理解がしやすくなります。LSTMの多くの説明では、各ゲートとtanh層の説明に注力するために、LSTMセルの図では、しばしば、隠れユニットのサイズについて省略されることがあります。実際の隠れユニットでは、各ゲートとtanhに重みとバイアスを持つ訓練可能なニューロンが存在します。したがって、概念的には隠れユニットのサイズが、モデルの表現の自由度を制御する役割を果たします。この役割は**モデルの容量**（**model capacity**）と呼ばれます。モデルの自由度とは、第3章のディープニューラルネットで説明しましたが、1つのモデルが表現可能なルールの複雑さを意味します。ユニット数が多ければより複雑なルールを学習できるようになります。隠れユニットのサイズは、隠れステートのベクトルのサイズ、セルのステートのベクトルのサイズ、重みの行列のサイズ、およびバイアスのベクトルのサイズを決定します。これらのことを念頭において以降の図を参照してみてください。

隠れユニットのイメージは以下のようになります。以下の□内の○の部分が隠れユニット数を示し、この図の場合、隠れユニット数は5として表現しています。

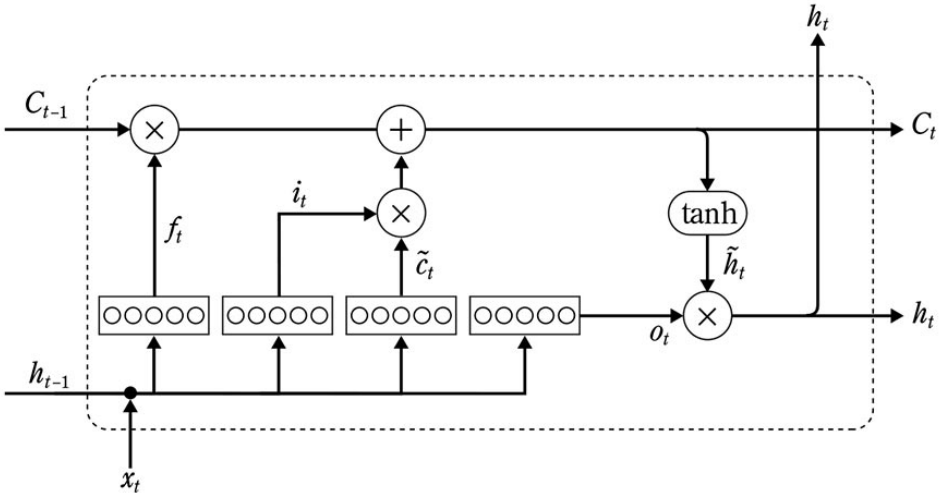


図5-13:LSTMセル内の隠れユニットのイメージ

次の図5-14内の数式は忘却ゲートの処理です。数式の下の図が各数式の処理データもしくは重み、バイアスの次元を示しています。忘却ゲートの出力のf_t、1時間ステップ前の隠れステート、バイアスは、それぞれ隠れステートのサイズと同じで、要素数が5個のベクトルになります。

ちなみに、これ以降の図の説明では、隠れステートのサイズは5、x_tの入力サイズは3にしています。

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

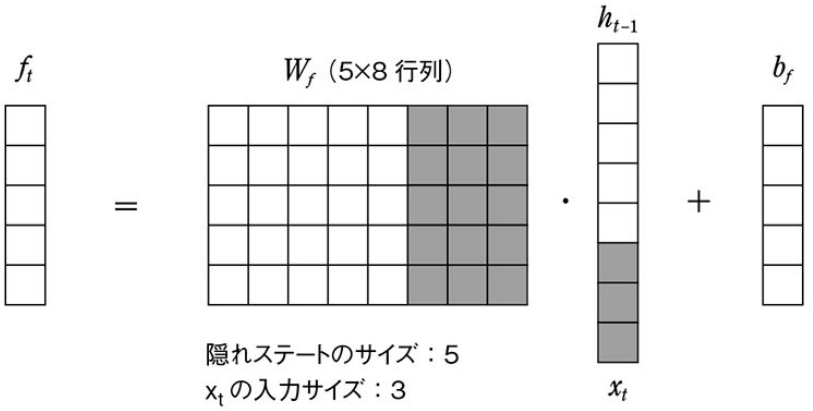


図5-14：［第1ステップ：忘却ゲート処理］の次元のイメージ

この図でグレーと白に分けた部分の意味についてこれから説明します。

図のW_f・$[h_{t-1}, x_t]$の計算は具体的には、以下のようなニューラルネットワーク構成で実現されています。図5-15の下段の［1 2 3 4 5 6 7 8］は、忘却ゲートへの入力データです。そのうちの［1 2 3 4 5］の部分がh_{t-1}のデータで、先ほどの説明のとおり、隠れステートのサイズにあたるデータ数は5つになります。［6 7 8］はx_tの入力データになります。

さらに、忘却ゲートのユニット数も隠れステートのサイズと同じ5なので、重みW_fは［忘却ゲートのユニット数］×［忘却ゲートの入力データ数］つまり5行8列になります。

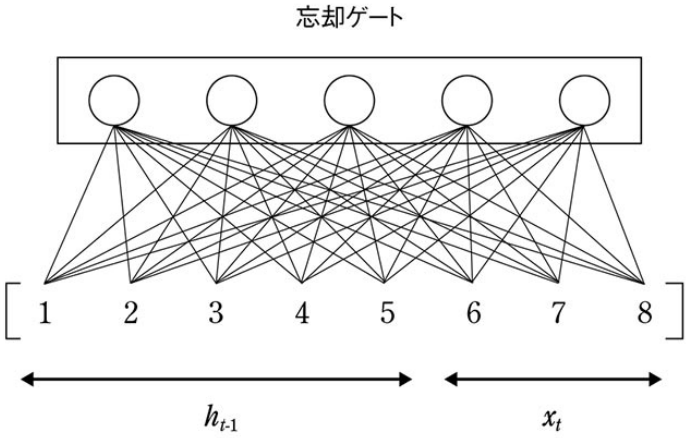


図5-15:忘却ゲートへの入力データ

図5-16では、図内の数式が入力ゲートの処理であり、入力ゲートの形は基本的に忘却ゲートとまったく同じであることが理解できます。そのため、データ次元のイメージおよびその説明も忘却ゲートと同じです。この図でも忘却ゲートと同じ次元であることがわかります。

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

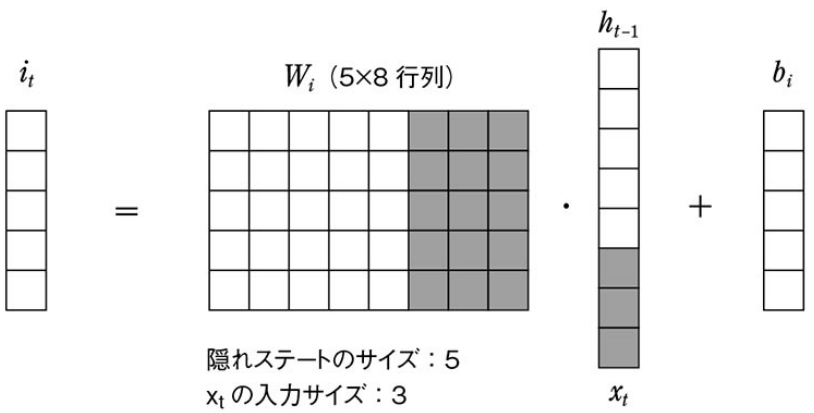


図5-16:［第2ステップ:入力ゲート］入力ゲート処理の次元のイメージ

次に、tanhでの活性化の処理におけるデータの次元について示します。こちらも活性化関数がシグモイドからtanh（ハイパボリックタンジェント）に変わっただけで、その点を除けば式の形がまったく同じであることが理解できます。

$$\tilde{c}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

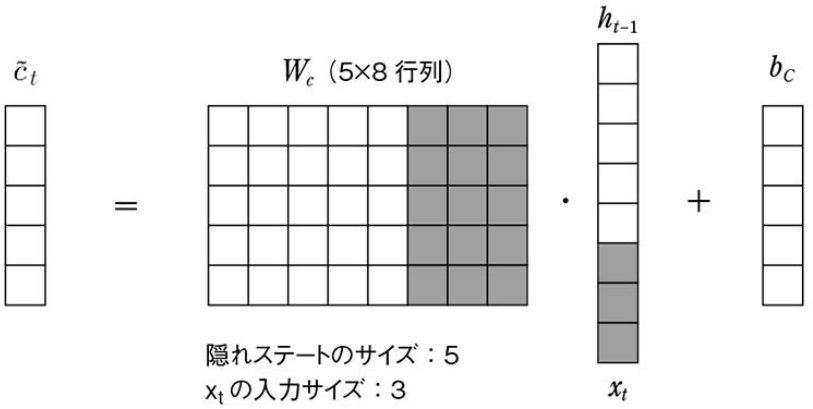


図5-17：[第2ステップ：入力ゲート] 入力ゲート処理におけるデータの次元

以下の図5-18では、すべて5個のデータ要素を持つベクトルで、同じ形状をしており、この図のような処理が行われます。

$$C_t = f_t * C_{t-1} + i_t * \tilde{c}_t$$

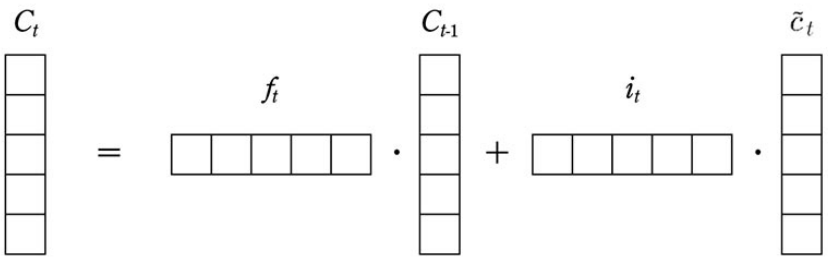


図5-18：[第3ステップ：メモリセル] メモリセルの処理におけるデータの次元

この後の出力ゲートの数式も忘却ゲートと同じ形をしているので、データ次元のイメージは変わりません。

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

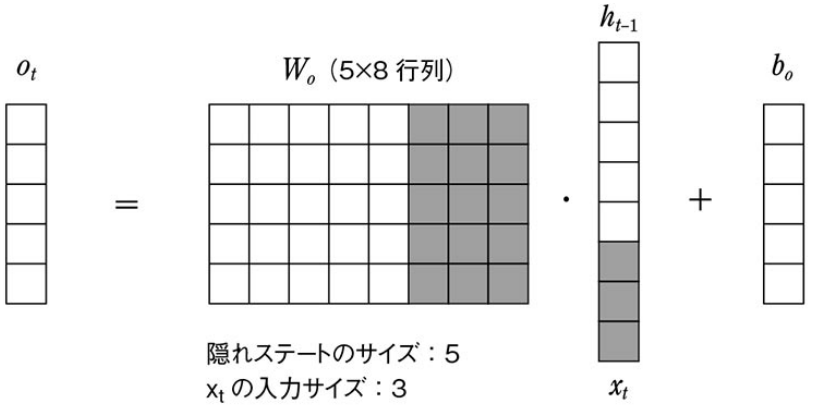


図5-19:［第4ステップ: 出力ゲート］出力ゲートの処理におけるデータの次元

また、最後に隠れステート処理におけるデータの次元についても触れておきます。この場合のデータの次元を表す図は、以下のようになります。非常にシンプルな式なので、次元のイメージもしやすいでしょう。

$$h_t = o_t \cdot \tanh(C_t)$$

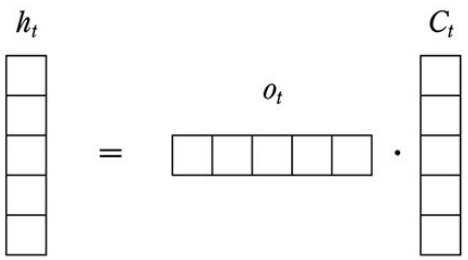


図5-20:隠れステート処理におけるデータの次元

各ステップのデータ次元の説明は以上です。

ここで、これまでのデータ次元の話を簡単にまとめておきます。x_tは、入力データの次元のベクトルです。たとえば、word embeddingの特徴が1000次元で表現可能な場合には、1000次元になります。

h_{t-1}、h_t、f_t、C_{t-1}、C_t、o_tは、いずれも隠れステートのベクトルと同じサイズであり、そのサイズと同じ次元数のベクトルになります。たとえば、隠れステートのサ

イズが128の場合には、それぞれ128次元のベクトルになります。

また、バイアスを意味するb_i、b_f、b_c、b_oも隠れステートのサイズと同じ次元数のベクトルになります。

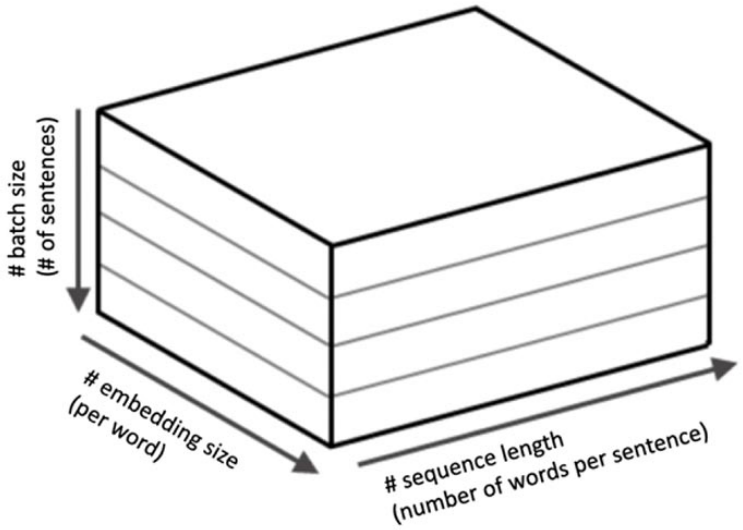
重みのW_f、W_i、W_c、W_oは、[隠れステートのサイズ]×([隠れステートのサイズ]+[入力データの次元])の行列になります。たとえば、隠れステートのサイズが5で、入力データの次元が3の場合、W_f、W_i、W_c、W_oは5×(5+3)つまり5行8列の行列になります。

ちなみに、隠れステートのサイズはすべての時間ステップで同じで、入力データの次元も同じであることに注意してください。

▶5.3.3.3 ミニバッチの次元の設定

ここまでは、例として単一のサンプル、つまり入力が単文であるような場合を説明しました。ただし、LSTMのネットワークを訓練するとき、入力が複数の文の集合になるようなミニバッチで最適化するのが一般的です。ミニバッチで訓練を行う場合には、今までに説明したデータの次元の考え方からどのように変わるでしょうか。今までの説明では入力データは1つの文であり、ベクトルつまり1次元のテンソルでしたが、複数の文を扱う場合には、入力が2次元のテンソル(または行列)になっただけです。それがもし3次元以上の行列だった場合には、3次元以上のテンソルになるということです。

たとえば、LSTMへの入力について考えてみましょう。上で見てきたように、時間ステップtでのLSTMセルへの入力は、先のコラムで説明したword embeddingの手法を用いて変換したベクトルになります。そして(単文の)1つのサンプルが、ベクトルの集合つまり行列になります。入力としてのミニバッチは複数の(文の)サンプルを有するように設定されているため、行列を連結させることで入力が立方体または3次元のテンソルになります。それを可視化した図としては、次の図を参照してください。



[出典]DL4J docs LSTM「Recurrent Neural Network」(https://deeplearning4j.org/docs/latest/deeplearning4j-nn-recurrent)

それぞれの文の長さは異なる可能性があるので、サイズの不一致を気にせずに単純に(行列となる)文を積み上げていけるのを不思議に思われるかもしれません。確かに何も処理を行わずに積み上げてしまうと当然可変長になってしまいます。そこで、ミニバッチの訓練では、各サンプル(文)のサイズを一致させるために、ミニバッチを同じ固定長になるように調整することが一般的に行われます。多くの場合、データセットのシーケンスの長さの分布について最初に分析を行った後、すべてのサンプルを合わせる必要がある最大長を決定します。決定した固定長よりもサンプルが短い場合には不足分を0で埋めたり、長い場合には切り詰めたりすることにより、それぞれのデータ長を同じ固定長に変換しています。

RNNの説明は以上になります。最後にRNNを直感的に理解するために非常に優れたアンドレ・カルパシー(Andrej Karpathy)氏の有名なブログ投稿を紹介します※4。RNNにバリエーションがあることについてはすでに触れましたが、その説明は彼のブログを参考にしました。また、彼は不連続データ、つまり画像の

※4 Andrej Karpathy:http://karpathy.github.io/2015/05/21/rnn-effectiveness/

シーケンスモデリングのスキームの適用方法についても触れています。実際、彼のブログを読むことで多くのエンタープライズのアプリケーションは、RNNが他のニューラルアーキテクチャと組み合わされることで、より多くのユースケースに対応できるようになることが理解できます。彼はブログの後半で、内部で起こっているニューロンの発火を理解するために優れた可視化について深い洞察を提供しています。この説明により、ネットワークがテキストのパターンを理解する方法を、ある程度直感的にとらえることができます。以上のようなテーマにご興味がある方はぜひ彼のブログを読まれることをお勧めします。

5.4 その他のトピックス

本章の冒頭から今までの説明で、RNNおよびLSTMの概要と主な機能がより直感的にイメージできるようになったでしょうか。この節では、RNNを使う際に知っておくとよいポイントを紹介します。

▶5.4.1 RNNのバリエーションについて

RNNは、取り組む課題の違いにより、ネットワークのアーキテクチャや、その中でも特にデータの入出力方法が異なる可能性があります。具体的には次の図5-21のようなバリエーションが存在します。

図の左から順番に説明していきます。なお次の図では、最下行の四角が「入力」、中央の行の四角が「RNNセル」、一番上の行の四角が「出力」を意味します。

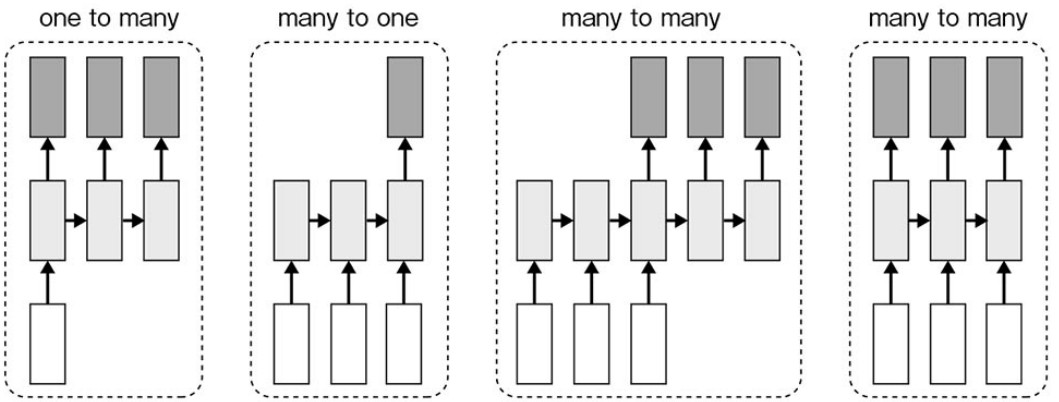


図5-21:RNNのバリエーション

●one to many:1つの入力データから複数の出力を得るモデル。

［例:画像キャプションの生成］この例では、1枚の画像からその画像のシーンを説明する文章などを作成します。

●many to one:シーケンシャルデータから1つの出力を得るモデル。

［例:感情分析］この例では、ある文章が好意的な発言か否定的な発言かを判断するような二値判断を行います。

●many to many(タイプ1):シーケンシャルデータからRNNを用いて一度特徴量を抽出し、その特徴量を入力として別のRNNでシーケンシャルデータの出力を得るモデル。

［例:機械翻訳］この例では、英語のシーケンシャルデータからその文の特徴量を抽出して、その特徴量をもとに日本語の文章として新たなシーケンシャルデータを出力します。

余談になりますが、RNNの自動翻訳では、たとえば、英語から日本語への変換の場合、英語の単語ベクトルから日本語の単語ベクトルに1種類のRNNを用いて変換するのではなく、一度、ニューラルネットワークで理解しやすい効率的な隠れベクトルと呼ばれる形に変換を行い、次に別のRNNを用いて隠れベクトルから日本語の単語ベクトルに変換する方式をとるのが一般的です。

つまり英文から隠れベクトルにエンコードしてから、日本語にデコードを行うという仕組みです。

このような仕組みにより、英語からフランス語など他の言語への翻訳を行う場合でも、英語から隠れベクトルへのエンコードという基本的な機能は利用可能となり汎用性が高まります。

●many to many（タイプ2）：シーケンシャルデータから異なるシーケンシャルデータを出力するモデル。

［例：単語の品詞のタグ付け］この例では、それぞれの入力である単語に対してそれぞれ品詞のタグを出力します。

読者の一部の方々は、シーケンシャルデータを扱うRNNのモデルを用いて分類や回帰のような予測がどのように行われるか疑問に思うかもしれません。上記のバリエーションからわかるように、それぞれのデータの出力タイプの違いにより、いくつまでの隠れステートの状態を出力として利用するかは異なります。それぞれの時間ステップもしくは最後の時間ステップで出力される隠れステートを入力として、まずFlatten層（フラット層）で1次元のベクトルに変換します。入力データが1次元のベクトルに変換できれば、全結合層へ結合した後、第3章で説明したとおり、解くべき課題が回帰か分類かの違いにより適切な活性化関数、損失関数や最適化のオプションを指定することで課題を解くことが可能です。たとえば多クラス分類のケースでは、最終出力として分類用のソフトマックスの活性化関数を指定することになります。

5.5 まとめ

本章を通じて、RNNが時系列や文章データに代表されるシーケンシャルデータを分析するのに適したニューラルネットワークモデルであり、その中でも勾配消失の問題などを解消できるLSTMというモデルが利用されることを理解できました。

また、シーケンシャルデータを分析するには、それぞれのデータの前後関係を理解することが重要なので、RNNではそれらの関係性を分析できるようにするために、前後の関係性の情報を保持するメモリ機能が実装されていることが重要であることを理解できました。

本章までがディープラーニングのそれぞれのアーキテクチャへの洞察を深めるための説明になります。では最終章で、実際のユースケースを想定したAIの開発の進め方の理解を深めていきます。

第6章

AI開発テンプレート適用のユースケース

機械学習をビジネスで利用するために

Deep Learning Project Template

機械学習による価値の創出に成功するには、顧客が実際にサービスで利用しているアプリケーションがあることに加えて、十分なデータがすでにあるまたは入手できる必要があります。この両方の条件を満たす課題を見つけ出すことは、機械学習の実装を含むプロジェクトが未経験だったり、アプリケーションの実装に不慣れだったりする方にとっては難しいでしょう。本章では、多くの企業が初めて機械学習に課題を適用する際に試すべき事項について、サンプルの課題を用いて説明を進めていきます。

6.1 ビジネスのポイントと分析

顧客の行動を理解するために分析を駆使する場合、ほとんどすべての人が何らかの形で取り組むべき共通の課題があります。まず分析を実行する人は通常、顧客の行動や習慣についてある程度の知見をベースに分析を行っています。同様に、機械学習の場合でも、分析を行う前にビジネスの典型的なポイントを洞察し理解することから始める必要があります。

ビジネスのポイントについて説明を行う際には、通常はビジネスに関する特徴や傾向を可視化するためのデータとして、SQLデータベースまたはExcelのスプレッドシートなどを用いることが多くなっています。このようなデータを扱うことにより、時間経過に伴う利益や、Webサイトの訪問者数といったビジネスのポイントを、端的に示せるようになります。

一般的に、収益のような単純なビジネス指標を可視化する方法について知識

を持っているチームは、Excelを使って回帰分析のジョブを実行することさえあるかもしれません。さらに一歩進んで、何らかの形でデータ駆動型の意思決定を行うために、機械学習をすでに適用しているチームもあるでしょう。

もし、あなたの所属する企業が、データ駆動型の意思決定を行うための準備がすでにできているか否かがわからない場合、1つのチームでシンプルな概念実証（Proof of Concept）を実施することから始めましょう。多くの場合、チームには解決すべきシンプルな課題が存在しているはずです。実際のチームの課題としては、全社的な視点での機械学習の分析を採用する前に、会社内の、ある1つの製品部門の中の1つのチームのような規模のユースケースに対して、機械学習の分析を試すケースが多くなっています。

一般的に、同じ製品部門であってもチームが異なれば、ITスキルのレベルは異なります。できるだけITのリテラシーの高い洗練されたチームを探して、小規模で制御が効きやすい分析を行います。このような分析の実行は、機械学習では**Experience（実験）**と呼ばれることが多いので、以降では「実験」という用語で呼ぶことにします。制御が効きやすい実験としては、通常、研究開発に時間を費やすだけの価値を有するROI（投資利益率）のユースケースから選ぶことが多いです。チームとしては、ある程度短期で成果を出せそうなシンプルなユースケースであるだけでなく、さらに、時間をかけてでも取り組むべき価値のある課題を選択する必要があります。この研究開発段階の主な目的は、チームが「実際に機械学習を組織に展開していくために何が必要なのか」について学習することです。

チームに、上記のようなビジネス指標を設定するのに必要な知識やノウハウが整っているかがわからない場合には、組織が抱えている特定の課題に焦点を当て、チーム間で横断的にワーキンググループを編成することをお勧めします。横断的なワーキンググループを構成する際には、課題と専門知識が各チーム間である程度オーバーラップするようにしたうえで、各チームが協力することで利益を得られるように、組織や人材を選定するのが望ましい方法です。そのようなプロジェクトに関心のある人材をピックアップするために、内部調査を行うことは有効で

す。もしくは、このような新しいプロジェクトに自ら手を挙げて参画してくれるような人材がいるのであれば、通常、より熱意があり成功に向けて一所懸命努力してくれる傾向にある、そのような人材を登用するのが好ましいです。

6.2 機械学習に対応できるチームやプロジェクトの選定

一般的に言えば、企業やチームは、機械学習を用いた分析の準備がいつ完了するのかについて明確に判断できる手段を持ち合わせてはいません。機械学習を用いた分析が実施できる段階になったことを正しく判断できるようにするには、以下のような事柄について考慮する必要があります。

1. チームにデータを保存するためのITインフラがすでに整備されていること
2. 通常業務としてソフトウェアをデプロイできる環境にあること。チームとしてデジタルトランスフォーメーションが完了していない場合には、一般的にまだ機械学習を用いる時期に達していないでしょう
3. チームはすでにある程度のデータ駆動型（データドリブン）の組織になっていること。具体的には、チームは収益やユーザの行動などについて、グラフなどのいくつかの指標を使って傾向分析する方法を知っていて、それらの情報に基づき意思決定ができる組織になっている必要があります

機械学習プロジェクトの成果が、常に何らかの形でデータ駆動型の意思決定に利用されるように習慣化してください。

◉――初めてデータサイエンティストを雇うために（既存のチームに組み入れる）

初めてデータサイエンティストを雇う場合、困難に直面される部署もあるでしょう。もしかしたら本当はその必要がない部門かもしれません。一般的にデータサイエンティストは、機械学習の技術を習得しているだけでなく、ビジネスアナリストの

知識も必要になります。ビジネスアナリストとデータサイエンティストの主な違いは何でしょうか。ビジネスアナリストの業務は、TableauやPowerBIのようなセルフサービスの分析ツールを使って収集されたデータセットの検証／可視化を行ったり、統計的な分析を行ったりすることです。それに対して、データサイエンティストの主な業務としては、ビッグデータの組成を明らかにし、ディープラーニングをはじめとする機械学習の手法も駆使して分析を行うことになります。ただし、データサイエンティストにとっても分析のさまざまな手法を学んでいくには、ベースとなる特定分野（ドメイン）の知識が重要となるため、社内のリソースからデータサイエンティストを探すのが理想的です。

第2章で説明したとおり、筆者の経験上、多くの企業で、機械学習の分析で失敗してしまう理由は主に3つあります。第1にデータサイエンティストを外部から採用すること、第2にデータサイエンティストにデータウェアハウスやデータパイプライン構築のような誤ったタスクを与えてしまうこと、第3にデータサイエンティスト専用の組織を立ち上げて組織全体の課題を彼らに解決させようとすることが挙げられます。これらはたいていの場合、うまくいきません。

多くの企業にとって、データ分析はあらゆる機械学習を用いた分析プロジェクトであり、最初に行うべきタスクです。PythonやRといった言語は、アナリストにとって非常に親しみやすい言語で、他の一部の言語のように特有のソフトウェアエンジニアリングの知識を必要としません。

人材採用の権限を持たれている方が外部からデータサイエンティストを雇用する場合に注意すべき点は、解決したい課題などを想定し、その課題を解くために必要となる特定の業種に関するドメイン知識を考慮することができて、チームを編成できるような人材を探すことです。データサイエンティストを雇って、彼らだけでデータサイエンスのグループ組織を作ったりしないでください。

データサイエンティストが実際の課題に対して集中して最短で解決してもらうようにする場合、面接のプロセスでは、基本的な分析に必要なスキルの知識と、ビジネスに関する知識を結び付けて適性を見るのが好ましい方法です。そのため、

理想的には、最初のデータサイエンティストは社内で調達するほうが賢明だと言えます。なぜなら社内の人間は、すでに必要となるビジネスに関する基礎知識を持っているからです。機械学習については学習用のさまざまな情報がオンラインや書籍などで入手できるため、自ら学んでいくことが可能です。

データサイエンティストを初めて面接する場合には、分析の基本にあたるSQL、ビジネスインテリジェンスの知識に加えて、新たなことを学んでいける能力があるかに注意してください。多くのデータサイエンティストは、採用者が求めるような特定業種のドメイン知識についてはあらかじめ持ち合わせていないので、自ら学んでいく必要があります。そのため、データサイエンティストが採用後のROIの指標を最短で達成するには、データサイエンティストがドメイン知識を有する既存のチームメンバーとうまく機能するかを確認してください。

6.3 課題の例：解約予測（Churn Prediction）

解約予測のゴールは、顧客がサービスのサブスクリプションを解約する時期をできるだけ正しく予測することにあります。一般的に企業は、解約予測により、サービスの契約維持に役立つ取引を積極的に顧客に送信するタイミングを把握しようとします。通常、解約予測は、一定数の顧客に対して割引情報を送信する際のマーケティング予算と対になっており、その割引がなければ顧客によってサービスが解約される可能性があることを想定しています。さらに顧客をできるだけ確実に維持するために、割引情報と合わせて、アンケートなどの調査の送信も有効になります。特に、サービスを終了しようとしている顧客は、企業からすると最も密にコミュニケーションを取りたい顧客対象となります。

顧客の解約や退会を事前に正しく予測することは困難ですが、顧客の定着率は、解約に至るか否かを判断するための情報として役立ちます。また、製品などの定期的な購買が行われるケースでは、製品の品質や性能に関する評価の情報なども、解約に至るかどうかを判断するうえで有用です。解約予測モデルでは、

新しい機能やサービスを追加した後で、サービス追加の影響について定着率などを繰り返しモニタリングして把握することで、サービスの調整や更新を繰り返し行っていく必要があります。また、解約予測では、顧客のニーズや顧客の属性ごとに**セグメンテーション**と呼ばれるグルーピングを行い、顧客セグメントごとに顧客の解約モデルを構築することもできます。

顧客のセグメンテーションを行う場合には、提供されるサービスに関して異なる価値観を持ち、行動が異なるグループをそれぞれ特定し、各グループのユーザが満たすべき条件を見極める必要があります。簡単な例として、ゲームのコミュニティを考えます。ゲームのコミュニティの場合、おおまかに以下の2つのグループに分けられます。1つ目のグループは、ゲームに多額の費用をつぎ込む、主な収入源となるヘビーユーザで、もう1つのグループは、お金にならない無料での利用をメインとする大多数のユーザです。顧客のセグメンテーションは解約予測に似た性質を持ちますが、それぞれのグループの違いを理解することが目的であるため、細かく分類すると解約予測モデルとは別の機械学習モデルになります。本章では話を単純化するために、顧客のセグメンテーションの詳細については触れませんが、顧客分析を行う場合の重要な要素の1つであることを覚えておいてください。

ただし、解約予測の精度を上げていくには、顧客のセグメンテーションも重要です。なぜなら、異なるセグメントの顧客はニーズが異なるため、サービスを解約してしまう理由がセグメンテーションによって異なる可能性があるからです。顧客のセグメンテーションは、ユーザを獲得してから特定期間の訪問ユーザの行動を指標化し、**コホート分析**と呼ばれる、ユーザをグループごとに分類しそれぞれの行動や定着率の分析をするところから始まります。そののち、どのようなプランに対して顧客が費用を支払うかといった標準的なセグメンテーションや、サービスを利用している企業やグループの規模などを含めたさまざまな顧客関連情報などを含め、さまざまな形で顧客のセグメンテーションをしていきます。チームや会社が異なれば当然セグメンテーションの目的やユーザが満たすべき要件もさまざまであり、各

セグメントの価値を最大化するためにはそれぞれの要件に対処していく必要があります。

6.4 解約予測で考慮すべき内容

解約予測の目標は、基本的にサービスを解約しようとしていると思われる顧客が、解約するか否かを“Yes”もしくは“No”の二択の回答を用いて正確に当てることにあります。

これは典型的な分類問題です。本書の第3章で説明したとおり、分類は、一連の既知の入力サンプルで学習したモデルを利用し、未知のデータに対して出力ラベル（今回の解約予測では“Yes”か“No”）を的確に与えていくことを意味します。説明を簡素化するため、顧客Aと顧客Bの2人がいる場合について考えましょう。今回解くべき課題は、顧客の属性や顧客の行動パターンなどの情報を収集し、そのデータに基づき解約するか否かを“Yes”、“No”のいずれかの回答で正確に予測するというものです。この分析では、過去にこのサービスを解約した顧客のデータセットを取得して、それらの顧客の各属性をマッピングすることで、今回の顧客Aおよび顧客Bがどちらのラベルに対応づけられるかを特定します。

既存の顧客は現在もサービスを利用しているため、出力ラベルとしては“No”の状態です。逆に、サービスをすでに解約している顧客は出力ラベルが“Yes”の状態です。我々の目的は、現在サービスを利用している顧客の中で、誰が将来サービスを解約してしまうかについてできるだけ正確に予測することです。予測を行うために使用できる分析方法はいくつもありますが、直感的には、機械学習により、データから顧客の属性などのパターン認識を行うことで分析できそうです。具体的な手順としてはまず、解約済みの顧客からその属性などのパターンを見つけること、そして、まだサービスを利用している顧客の中から、解約済み顧客と似た属性の顧客を洗い出すことです。これにより、将来サービスを解約する可能性の高い顧客に“解約可能性あり”といったラベルを付けるか、もしくはその顧客が解約

する可能性について、確率を計算します。重要なことは、できるだけ早く解約を阻止することです。解約しそうな顧客の属性を予測する際には、顧客の中から、解約につながる顧客として区別可能な属性の中で、顕著な要因を見つけ出すことが重要です。解約予測をする際に、区別に役立たない属性を用いて分析を行ってしまうと、予測精度が落ちたり、分析のスピードが遅くなったりする原因につながるため、役に立たない属性はできるだけ排除すべきです。

解約予測を行う場合、一般的に二値の分類問題を扱うことになります。二値分類のモデルをチューニングして未知のデータに適用するには、第3章ですでに触れたとおり、二値の分類問題特有の考慮すべき項目があるので、その点は注意してください。

YesかNo の二択問題ではしばしば、特定のリスクの特性が関連付けられます。YesかNoの二択にする際には、最終的に2つの値のうち、いずれか1つの値に結論を寄せることになります。そのため、ある程度高めの損失が生じるリスクがあります。解約予測の問題と不正検出の問題では、不正検出のモデルのほうが一般的に解約予測の問題のモデルより検出するのにより多くの費用や労力がかかり、実際は陰性であるにも関わらず誤って陽性と判断してしまう偽陽性のケースも多くなります。解約予測のモデルでは、偽陽性の検出にかかる費用や労力は、たとえば解約する可能性の高いある顧客に対して無料サービスもしくは割引情報をメールすることで確かめられるので、決められた予算枠の費用と労力の範囲で実施できるという特徴があります。それに対し、不正検出の場合、たとえば銀行の取引のケースでは、不正の可能性のある取引に対してタグ付けを行う形で検証を行っていきます。取引によって取引額は異なりますし、その取引の真偽を特定していくのに手間、さらには時間も要します。たとえば店頭でその顧客の金融取引の了承をすぐに通さなければならないケースでは、取引の真偽を特定できるまでの間、顧客にも待つことを強いるようになります。返信までに時間を要してしまうと、そのせいでイライラした顧客はサービスを解約してしまうかもしれません。 このように顧客のサービスの利用シーンに応じて解約につながる要因が変わるため、利用

シーンも含めた顧客のサービス全体についても正しく理解しておくことが重要です。

解約予測では、かなり高い精度で安定してYesかNoを選択するのは難しく、顧客のセグメントや業界の特性によって精度は異なります。それぞれの特性に応じて適切な精度を設定することも重要になります。そのためにも十分なドメイン知識を有するメンバーが本プロジェクトに参画して、適切なアドバイスを与えることができるように環境作りをしておくことが重要になります。

次に、解約予測の最善の方法を考える際の基本的な手順について説明します。

6.5 解約予測で用いる探索的データ分析

探索的データ分析（EDA：Exploratory Data Analysis）は、機械学習の開発ライフサイクルの中で、データの傾向や属性について探索的に理解するステップとして定義されます。解約予測モデルを構築する際にEDAを実施する目的は、顧客のさまざまな属性などを調査することにより、異なる顧客の種類をできるだけ正確に区別できるようにすることです。顧客の属性の種類はさまざまです。特定のプランのサービスを考えた場合、通常、顧客によるサービス申し込み後のすべての顧客行動に関する情報などが顧客の属性データに含まれています。

EDAを実行するときには、顧客の属性のうち区別可能な要因を探す必要があります。第2章で主成分分析（PCA）について解説した際、分散の変動要因にはトレードオフがあることを説明しました。分散が重要である理由は、データ内の属性で分散が大きいほど、モデルは異なる種類の顧客を分離しやすくなり、価値のある分離パターンを導き出すことが容易になるためです。

顧客の属性を区分するのに機械学習のモデルが利用される理由は、機械学習のモデルでは、常にさまざまな種類のサンプルを分離して区別し、有効な分離パターンが見つかるまで分離を繰り返していくことが基本になるからです。属性の

変動が大きい場合には、それぞれのデータサンプルに冗長性が少ないことを意味します。一般的に分散が低い変数はモデルから削除しておく必要があり、EDAのステップでそれらの変数を削除するようなスクリーニングが行われます。

6.6 最初の機械学習の実験（Experiment）

最初の実験は、EDAステップの後で実施する必要があります。実験の準備段階では、実験で用いるシンプルなモデルとそのモデルを構築するためのツールの選択を行います。選択されるツールは、一般的にチームによって異なります。一部のチームはJava、Scalaなど特定のプログラミング言語を利用するかもしれませんが、一般的に機械学習を利用するチームではRもしくはPythonのプログラミング言語が用いられるでしょう。

解約予測を行う場合には、シンプルなモデルとして、ロジスティック回帰やランダムフォレストなどが一般的に利用されます。これらのアルゴリズムが選択される理由は、高速に学習を行うことができ、メモリの使用量も低く抑えられる点にあります。これらのアルゴリズムであれば、自分の手持ちのノートパソコンでモデルを構築できますし、かなり小規模なサーバ上で実行してもすぐに結果を出すことができます。

ニューラルネットワークのように計算に時間を要するモデルを実験の最初で用いるのは、一般的にあまり賢い方法ではありません。第2章で説明したディープラーニングを用いるべきいくつかの条件に適合すると想定される場合には、まずはロジスティック回帰のアルゴリズムで妥当性を見ることをお勧めします。ロジスティック回帰はシンプルなニューラルネットワークの形をしているため、より複雑なニューラルネットワークよりも試しやすいからです。

用いるべきアルゴリズムによらず、EDAのステップでは多くの時間を使って洞察を深めるようにしてください。特にこの種の課題で精度を高めていく場合、分析に必要な入力データを正しく選択することが、最適なアルゴリズムを選択するより

も精度向上に効く場合が多いので、EDAに十分な時間を取ることが非常に重要です。アルゴリズムを選択する場合には、YesかNoの二択の問題（一般的な二値問題）に適したものを選んでください。この点については次の節でさらに詳しく説明していきます。

EDAの次は、モデルの性能を検証するフェーズになります。このフェーズでは、精度の確からしさを調べるための指標の選択と、データセットの分割割合の設定が必要です。プロジェクトで求められる精度に適した指標を選択する必要があります。第3章でどの精度の指標を用いるのかについて説明しましたが、精度を測定する指標として適切なものを選択するには、誤検出で生じるリスクとの関係性を考慮することが重要です。二値の分類で一般的に用いられる精度の評価指標は、**AUC**（Area under an ROC curve）です。AUCの値が1に近くなればなるほど誤検出が低くなり精度が上がります。二値の分類でAUCの一般的なしきい値は50%をデフォルト値として用いますが、必要に応じて課題に適した値になるまでしきい値を調整していきます。

データセットの分割についても第3章で簡単に触れましたが、全体のデータセットを、学習用とテスト用に7:3の割合で分割することをお勧めします。ちなみに、モデルを作成する際に、学習用とテスト用にデータを分割して評価する方法は、ホールドアウト法と呼ばれます。モデルでYesかNoの分類が正しくなされているのかを確認するために、モデルで一度も用いられていない未知のデータのサブセットを使ってモデルの精度を検証する必要があります。検証のためにホールドアウトのような方法を選ぶ際には、それぞれの分類クラスに対して精度を測定できるのに十分なデータサンプルが存在することを確認してください。できるだけサンプルデータは特定のクラスに偏ることなくバランスが取れているほうが望ましいです。このようにしておくことで、モデルはそれぞれのクラスを偏ることなく分類できるのみならず、それぞれのクラスからずれるような例についても正しく区別しやすくなります。

6.7 モデルの構築方法

前節では、モデルの検証方法と、解約予測の例にもとづく実験を設定する際の検討事項について説明しました。次にモデルを構築するためのメカニズムについて説明します。

ニューラルネットワーク、ロジスティック回帰、またはランダムフォレストのいずれかを使用して解約予測モデルを定義する場合、モデルを構築するために適切な処理フローを形成することが重要です。

機械学習を行うための一般的な処理フローは以下のようになります。

1. 数値計算を可能にするために、カテゴリデータが存在する場合には、カテゴリデータをone-hotエンコーディングのようなデータ変換手法を用いて数値データセットに変換すること
2. データの分散に偏りをなくすため、ゼロ平均単位分散を用いて浮動小数点値の正規化を行うこと
3. （処理フローの1番目のカテゴリデータの処理に似ていますが）True/FalseやYes/Noのような値についても0/1の数値に変換すること
4. 連続データなどを分類として扱いやすくかつ意味のある離散値データに変換すること

上記の処理フローは、分類のみならず回帰予測モデルでデータパイプラインを定義する際にも利用されます。ここでは、非常に基本的な解約予測の分析手法を想定しています。具体的には、データベースまたはExcelのスプレッドシートなどに格納された顧客属性データを使って、顧客のパターンを識別するようなケースを想定しています。

現場では今までの説明で触れていない、そのドメインならではのテクニックが存在するのが一般的です。顧客の感情や顧客取引をより精度良く知るために自

然言語処理（NLP）を用いる場合などで、ドメインならではのテクニックが発揮されます。ただし順番としては、最初に一般的な顧客属性などの基礎的なパターン認識を行った後、次のステップでより深くそのドメイン特有の顧客とのやり取りなどを知るためにNLPなどを用いることをお勧めします。

解約予測モデルでは、カテゴリ属性のうちYesかNo（True/False）にあたるブール代数の真理値の機能を活用して、顧客の属性を表現することがしばしばあります。その理由は、モデルが特定の属性だけに過度に依存しないようにするためです。

最後に、十分なデータ量（可能な場合は少なくとも数千のサンプルデータ）でモデルをトレーニングすることを忘れないでください。そうしないと、モデルは問題を適切に表現できないため、精度が十分に出ず、有用なモデルにならないからです。

6.8 モデルの展開

モデルを本番環境に展開（デプロイ）する場合、通常、データサイエンティストではなく、ソフトウェアエンジニアリングのチームが作業を行う必要があります。機械学習のモデル開発を行うデータサイエンティストはモデルの構築とチューニングを主な業務にしているため、あまり実際の本番の商用環境に触れる機会がありません。大規模なデータを活用するシステム基盤の構築や運用を行うソフトウェアエンジニアは、データエンジニアと呼ばれ、彼らが本番環境への実装の役割を果たします。そのため、モデルの構築からモデルの本番環境への適用、維持まですべての業務を1人でこなせるようなエンジニアはほとんどおらず、モデルの展開は、データエンジニアが分業として行うのが基本的な考え方になることを理解しておきましょう。

通常、モデルをサービスとして展開するツールとして、さまざまなサービスやオープンソースのプロジェクトが存在します。読者の皆さんの所属するチームがモ

デルの展開に慣れていない場合でも、これらのツールを使用することでモデルの展開を実現する際の障壁はかなり低くなります。アマゾンのAWSに代表されるように、モデルの展開と管理に重点を置き、SaaS（Software as a Service）と呼ばれるクラウドの形態で顧客が必要とするソフトウェアをインターネット経由で利用可能とするサービスを提供する企業は、多数存在しています。利用するサービスやプラットフォームを選択する場合、ベンダー固有の要件で後々縛られないよう、できるだけオープンソースベースの利用を推進し、ベンダーロックインにならないような環境を検討してください。それらのサービスのうち、読者が必要としている要件以外のものについても一緒に使える環境になっている場合は、できるだけ必要最低限のものに限定できる環境へ移行するようなパスを計画してください。これは、長期的にサービスの開発を行っていくうえで、毎回の作業をコンパクトにできるため、効率的な開発を実現するのに役立ちます。特に機械学習のように急速に発展している分野のサービスを利用する際には、単一ベンダーの提供環境に長期的に縛られない、つまりベンダーロックインにならないよう、細心の注意を払うことが重要です。

商用環境にモデルを展開する場合には、モデルが商用利用に可能な条件をクリアしているかを評価できるようにするために、モデルの展開に関して考慮すべき事項、特にプライバシーやデータ規制などのポリシーも含め、第2章で考慮した内容を理解しておいてください。

モデルは、YesやNoなどのシンプルな答えを返すように、WebシステムのHTTPで呼び出すREST APIとして展開されることが多くなっています。解約予測の場合、ある答えとなる確率を返すのではなく、YesやNoなどの単純な答えを出力する基本的なマイクロサービスを構築する必要があります。後処理としては、他のサービスでその答えを再利用されることが想定されます。そのため、容易に再利用できるようにマイクロサービスとして設計し、モデルを展開できるようにします。

モデルを展開するサービスを選択する前に、まずモデルが本番環境に実際に

適していることを確認してください。たとえば、モデル自体は正しく構築されても、商用環境に即した適切な分散化や、商用で使う大量データの処理を行うのに十分にスケーリングできていない場合があります。この場合、商用でトラブルに陥る可能性があります。プロジェクトごとで本番環境に求められる利用条件は異なるため、サービスを実装する前にそれらの利用条件を確認しておきましょう。

具体的な検討例を挙げましょう。たとえば、モデルの展開についてどの程度複雑にする必要があるかについて考えます。スタートアップ企業でプロジェクトを担当したとすると、アプリケーションは小規模で済むので、小規模な環境で動作しやすくするために、アプリケーションに機械学習モデルを直接組み込み使いこなせるようにするかもしれません。組み込みとは、モデルが利用されるアプリケーション内にモデルを実装することを意味します。通常、アプリケーションを展開するには、アプリケーションのソースファイルなどをひとまとまりにしたアーカイブ形式にして、モデル自体もアーカイブにバンドルします。アプリケーションのその他のソフトウェア・コンポーネントと組み合わせることで、運用・保守やバージョン管理を簡単に行えるようにする場合が多くなっています。一般的には、小規模であれば組み込みで対応するほうが運用や管理が楽になります。

解約予測以外のモデルでは、より多くの計算時間を必要とする可能性があり、汎用CPU系のハードウェア上に実装されたアプリケーションで商用環境を運用していくのが適さない場合があります。そのような場合には、より高速な計算を行えるGPUが実装されたハードウェア環境に展開を行う必要があります。モデルをできるだけマイクロサービスで展開するように推奨する理由は、より適切なハードウェアを含めた実行環境上でモデルを動かす必要があり、コンパクトなサイズのサービスにしておくことで、手軽に環境を選択して展開しやすくなるからです。

本番環境では、使用可能な同じモデルを複数コピーして利用することにより、負荷を分散して高可用性を確保する場合があります。その場合、モデルを含むマイクロサービスの堅牢性を担保するために、メンテナンスコストやサービス実装の複雑さを比較検討する必要があります。

解約予測の場合、モデルは通常それほど複雑ではなく、計算が集中することもありません。そのため、展開で考慮しなければならない検討事項としては、特殊なコンピューティング要件まで考慮する必要がなく、標準のソフトウェアエンジニアリングの知識の範囲で対応できます。

最初のモデル開発では、たとえばアーカイブのアプリケーション内にモデルを展開するために、可能な限り標準的なソフトウェアスタックにモデルを配置するように検討してください。

本番環境の標準的なソフトウェアスタックと異なるプログラミング言語でモデルが記述されていれば、モデルの書き換えが必要になる場合があります。このようなケースはかなり一般的です。企業が商用環境で使っているプログラミング言語はJavaであるのに対し、実験で用いられるプログラミング言語はPythonやRであるといった場合があります。これは、所属組織の製品チーム内で定着したスキルの違いによるものでもあります。組織で十分に浸透していない新しいプログラミング言語を採用すると、多くの場合、システムの展開や維持を行うデータエンジニアのチームの負担が増え、モデルの維持がますます複雑になります。モデルはPythonなどでプログラミングされるケースが多いでしょうから、最初のモデル開発から本番環境で利用するモデルの開発へ移行する際に、本番環境のプログラム言語に移植することをお勧めします。なぜならプログラミング言語の仕様の違いによる動作の不具合などを完全に回避できるからです。

ただし、使用している実行環境やユースケースによっては、移植を行う必要がない場合があるので注意してください。商用の環境と実験の開発環境で同じ言語が使われ、ソフトウェアスタックも基本的に同じものが使われている場合もあります。

機械学習のモデルを初めて展開する場合には、商用の環境との差分を常にできるだけなくすように心掛けてください。このことは最短で成果を上げるために非常に重要なポイントになります。新しい言語や仕組みを導入すればするほど、チームもしくは組織内で機械学習による分析を採用するのが難しくなってしまう点に

注意しましょう。

6.9 解約予測モデルの維持管理

モデルを展開してから、時間が経つにつれてモデルの精度が悪くなる可能性があります。そのため、精度の監視を常に行う仕組みを導入しておくことを忘れないでください。特に、提供しているサービスや製品が新しく更新されていくにつれて、顧客の行動も時間とともに変化しやすく、そのような進化は止まることはありません。少なくとも新サービスや新製品のリリースにあたっては、モデルを更新する準備を常に心掛けてください。モデルも、製品やサービスと同じく資産として維持管理を行う必要があることを覚えておいてください。精度が出ない本番環境のモデルを運用し続けることは、非常にリスクが高く、誤った予測結果により事業に悪影響を与えかねません。

解約予測では、顧客が解約する可能性があるサービスや製品の中で、特に新機能を実装した際の差分をできるだけ正確にとらえて予測することが重要です。多くの場合、顧客は新機能を好まず、解約につながりやすいという特徴があります。顧客が解約する可能性がある時期を測定するために、新たなバージョンのサービスおよび現在のバージョンのサービスを一定期間並行して利用してもらえる環境を整え、解約の可能性のある顧客からフィードバックを収集できるようにします。実際に、そのような仕組みを設けるなどの施策をとることが多くなっています。このようなフィードバックのループを解約予測モデルに組み込むことで、継続的に利用可能なレベルの精度を保てるようにする必要があります。

モデルを展開するときには、本番環境で高い品質を保つために、新しいモデルを展開する前に常に精度をチェックする必要があります。そのため、事前に定義されたテスト用のデータセットを準備してください。品質チェックのためのテストデータを用いて品質チェックを行う仕組みを、ソフトウェアチームによるモデルメンテナンスの通常業務の一部として組み込んでください。この厳格なテストセットを用い

て精度のチェックを行うことにより、チームは以前展開したモデルと比較できて、客観的な良し悪しを評価できるようになります。さらには、精度が出るまで修正を行うことが容易になります。

本番環境では、事前に予期不可能な事態が発生する可能性が常にあります。本番環境を安定して利用できるようにするには、モデルを自動で展開するパイプラインの中で、モデルの精度をモニタリングしたり、モデル自体が正しく動かないことを検知したりする機能を準備しておくことが非常に重要です。

6.10 長期的に価値を維持するためにすべきこと

解約予測の価値を長期的に維持していくには、1つのモデルを維持するだけではなく、新しいモデルを常に構築できるようにしておきましょう。

新たなモデルを継続的に構築できる環境を整えることにより、たとえば、1つのモデルで機能することが証明された場合、より具体的な課題に対してそのモデルを拡張して対応できるように検討することも容易になります。先に説明したように、ある1つの顧客のセグメントで有効性が証明されたモデルを別の顧客のセグメントの解約予測に当てはめる場合に、新たなモデルを継続的に構築できる環境を整えることは、有効な手段になります。機械学習による分析が継続的に維持され新たな価値を生み出し続けられる状態になったら、引き続き機械学習に対して投資を行い、チームの教育を強化していけるようになります。成功し続けていくためには、チーム内にデータサイエンティストを組み込んで、チーム内の他のソフトウェアエンジニアと協力して、価値あるモデルを最適な形で展開／運用できる体制を整えてください。

多くのモデルが継続的に展開できるようになったら、各プロジェクトの投資判断に関わるすべてのステークホルダーに対して、モデル実装の成果が正確かつ透過的に伝達されることを確認してください。このような仕組みを作ることで、組織は客観的にモデルに対する評価の基準を設けられるようになり、チームはそれら

の判断基準をもとに、プロジェクトに対して適切に投資しやすくなります。

データサイエンティストがチームに新たに参加したときには、ソフトウェアエンジニアリングとチームのドメイン領域の知識について理解してもらうことも忘れないでください。それぞれのチームでこのようなノウハウをまとめておき、ルール化していくようにしましょう。そうすることにより、各チームでは、そのチームと関わるデータサイエンティストが最短でチームの課題に取り組めるようになります。ひいては、データサイエンティストを新規採用などで増やしていく際にも、標準的なノウハウとしてルール化できるようになります。

組織全体で横断的なデータサイエンティストを抱える場合にありがちなのは、組織に真の価値を与えるのではなく、研究領域での実践にとどまってしまうことです。解約予測のように、一見すると非常に単純な問題から実践をスタートし、チームに組み込まれたデータサイエンティストが特定のサービスや製品のドメインを並行して学習していけるようにすることで、より多くの価値をすばやくチームにもたらせるようになります。

データサイエンティストをチーム内で抱えられるようになると、モデルの開発だけでなく、エンタープライズで利用可能かつインテリジェントなアプリケーションへの実装まで、チーム内で維持管理できるようになります。このチームの体制が長期的に維持管理できるようになると、維持管理に必要な知識が蓄積されると同時に、データサイエンティストとその他のソフトウェアエンジニアがともに1つのサービスをすばやく提供するための多くのノウハウを連携して迅速に学べるようになります。こうした状況を乗り越えれば、定常的に機械学習も利用できるチームとなります。それは組織にとって非常に大きい財産になっていきます。

組織内のエンジニアリングの役割という視点で見ると、一般的にデータサイエンティストも、多くの特権を持つように扱うべきではなく、組織内にいるさまざまな分野のソフトウェアエンジニアの1人として見なされるべきです。データサイエンティストのスキルもさまざまです。長年の経験からモデルのサービス展開についての知識を有する人もいれば、オンラインクラスやブートキャンプの知識を学んだだ

けで今後さらにより多くの知識を学ぶ必要がある人もいます。データサイエンティストを多数雇う前に、チーム内で必要とされているインテリジェントなアプリケーションを開発し維持管理するためには、必要となるさまざまなエンジニアをバランスよく雇うように心掛けてください。

もう1つの考慮すべき事項は、AIの展開のアプローチとそれを発展させていく際に必要となるインフラについてです。初期投資で成功した後は、チームがより複雑なモデルを構築したり、より複雑なインフラストラクチャ（複数のモデルを一度にデプロイするなど）が必要になったりすることがあります。そうした場合には、機械学習からさらなる価値を引き出すために、アプリケーションのアーキテクチャを再検討し、機械学習を用いた分析の価値に見合ったインフラを整えることが必要です。複数のモデルを同時に開発／展開していくような場合、エンジニアの視点では、それぞれに十分なコンピューティング環境をそろえたいと考えるでしょう。しかし、インフラ投資が大きくなりすぎると、維持管理コストが機械学習でもたらされる利益を相殺してしまう可能性があります。プロジェクトの責任者は、先に示した分析結果を透過的に伝達できる仕組みを使い、インフラ投資が適切なものであるかについて判断する必要があります。

6.11 従来の解約予測と機械学習による方法の比較

解約予測の目標は、顧客にサービスを使い続けてもらうことにあります。一般的な解約の予防策としては、従来のマニュアルのアプローチを使って実装することも可能です。企業で一般的に使われている従来の手法は、顧客の利用状況や顧客特性などをグラフで可視化し、その時間経過に伴う基本的な傾向を理解していくというものです。これは、マニュアルのビジネスインテリジェンス的なアプローチです。一般のビジネスアナリストは、顧客の特性を深く理解するために、データを注意深く調べることで、ある顧客が解約に至った根本的な要因を理解しようとします。これはビジネスアナリスト自身の知識に基づくルールの分析になりますが、

非常に重要なアプローチであるとともに現在でもとても有益なアプローチです。

それに対して、機械学習を用いた解約予測のアプローチでは、顧客が解約しそうなパターンを迅速かつダイナミックに浮かび上がらせることができます。

特にビッグデータのようにデータ量が非常に多く、顧客の特徴の情報（年齢、性別、収入など）が多ければ多いほど、ビジネスアナリストが主に人間の知能によってすべての特徴の相関性を見つけ出すのは至難のわざですが、機械学習では複雑な相関のパターンでも簡単に見つけることが可能です。つまり、データ分析をマニュアルで行う場合には適切にスケーリングすることができず、情報量等がある程度以上多くなると事実上マニュアルだけで分析を行うことは困難になります。解約予測では、大まかなトレンドを理解するだけでうまくいくこともあります。しかし、通常の企業ユーザが行う解約予測では、非常に細かいレベルで物事を理解することが必要になる場合が多く、データに含まれるノイズをうまく処理しながら複雑な顧客の行動の関連性を正確に理解していく必要があります。そのため、機械学習を用いて、大まかなデータの相関関係をとらえ、その結果をもとにビジネスアナリストがその相関関係の背後にある真の関係性の理解にたどり着く、というような手順を踏んでいるケースが多くなっています。

また、多くの顧客を抱えている企業の場合、マニュアルで分析して顧客のトレンドにすばやく対応することは困難なので、機械学習を用いて解約予測を行っている例も多数あります。

マニュアルによる分析も機械学習による分析も一長一短があるため、どちらが正解ということは特になく、用途に依存します。チームとしては、さまざまな用途で適切な分析を効率的に行えるように備えることが重要であり、機械学習を用いた分析も必須となっているため、機械学習の解約予測についても積極的に取り組んでいただくことをお勧めします。

6.12 おわりに

本書の目的は、ディープラーニングをはじめとする機械学習をうまく利用する方法について正しく理解することでした。機械学習がどのように適用されているか、そして数学をあまり深く掘り下げずに基本的な問題について洞察を深めていく方法を、読者が理解できるようにすることでした。数学は重要ですが、過度に数学に集中することは、機械学習に初めて触れる読者にとっても有害です。「機械学習をどのように適用するか」「どのようにプロジェクトを進めていけばよいか」について本書で知ることにより、機械学習の実践に成功することに一歩近づくことができます。さらには、実用に重きを置いて、機械学習の関連情報を見ることができるようになります。

読者は、「数学」「統計学」のほか、「日常のデータサイエンスや機械学習で使用される必要なソフトウェアエンジニアリングスキル」など、機械学習に関するより高度なスキルの獲得に時間を費やそうとするかもしれません。それ以上に著者が読者に期待するのは、AIという名の誇大広告のために機械学習を勉強したり実装を試みたりすることではなく、適切なビジネスシーンで利用できるだけの知識を獲得し、エンジニアリングの別の分野も組み合わせて機械学習の実装に取り組み始めようとすることです。組織が直面している実際の問題や、本質的に興味深い課題を見つけ、それらに対応するために機械学習を利用していくことを心掛けてください。

今回紹介した解約予測の例は、ほとんどの組織が抱える問題を非常に単純化して定義したものです。これは、画像を分析するコンピュータビジョンのような問題を反映したものではありません。コンピュータビジョンの問題は、解約予測で説明されるものとは異なり、一般的に多くの計算リソースが必要となる点です。

通常、解約予測モデルでは、必要な計算リソースは少ないので、一般の読者が手持ちのノートパソコンなどのCPUリソースを利用して、数分でモデルをトレーニングできます。このような機械学習モデルの開発や展開では、コンピュータビジョ

ンで必要となるような特別な知識は必要ありません。コンピュータビジョンでは通常、多くのワークロードを適切な速度で実行するためにGPUが必要であり、GPUで機械学習を動かすための知識が必要となります。したがって、最初のプロジェクトとしてコンピュータビジョンを選択するのは適切ではありません。

最初のプロジェクトを試した後は、すぐに次の機械学習プロジェクトでモデルを構築したり、組織にAIを導入したりするのではなく、チームに適切な専門知識と必要なデータやインフラストラクチャがあることを確認してください。これについては第2章で説明しましたが、AIのユースケースでは、チーム内だけでは十分に精通していない項目についても決定しなければならない場合があることを理解してください。このような場合には、それぞれの項目で適切に対応できる知識を有するチームなどにアドバイスを求めるようにしてください。たとえば既存のチームでGPUなどの新しいチップを採用する場合には、大きな変更が必要になります。そのような場合には、GPUの採用実績があるチームからのアドバイスに従い、機械学習のモデルなどをタイムリーに適切に実装できるようにしてください。そのような変更を容易にするために、サービスの開発を行う場合には、大規模なサービスをひとまとまりにして構築する方式ではなく、複数のマイクロサービスを組み合わせることで展開できるようにして準備しておくことをお勧めします。

従来型のIT組織では、機械学習の採用に時間がかかり、機械学習を採用することで生じる新しい複雑さのすべてに対処する準備ができていない可能性があります。大きなリスクには大きな見返りがあります。本章の前半で述べたように、チームは機械学習を開始する前に、デジタルトランスフォーメーションのイニシアチブをすでに経験している必要があります。チームがデータを保存したり、ソフトウェアを効果的に展開したりできない場合、機械学習を試みることはお勧めできません。ただし、組織がITの標準化を進めていくためのインセンティブとして、機械学習に取り組むというケースがあるかもしれません。そのような場合には、個人が機械学習のユースケースのアイデアを自宅で軽く試してもよいものと思われます。つまり、デジタルトランスフォーメーションが前提でなくても機械学習を使用し始めて

かまいません。この場合、できるだけシンプルで、比較的簡単に結果を出せるようなプロジェクトを選び、これまで説明したシンプルな手順を実行することで、成果に結び付けることが可能です。その点についても理解しておいてください。

本書の内容はここまでです。最後までお付き合いいただき、ありがとうございます。本書では可能な限り数式を用いずに機械学習のいくつかの概念を直感的に理解できるように説明し、さらには機械学習を用いてプロジェクトを推進していくために必要な手順と進め方についての考え方をカバーしました。本書を通じて、今後、読者の方々が一人でも多く機械学習の分野で活躍されることを願っています。

Appendix

AIテンプレートを実装したサンプル

Docker上で動作する機械学習パイプライン

Deep Learning Project Template

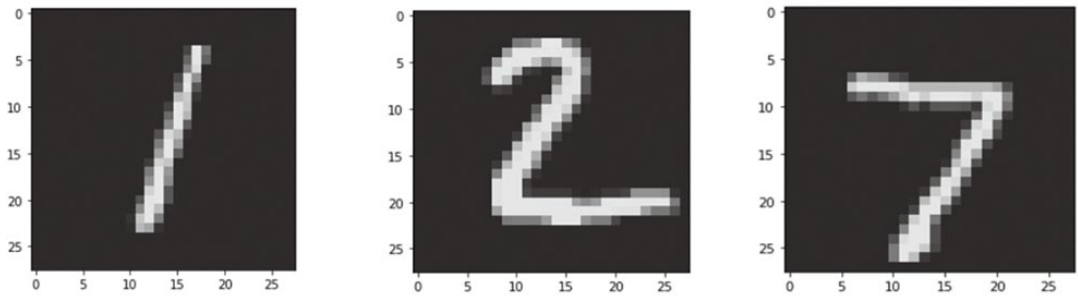
ここでは、第2章で紹介したAIテンプレートの概念をさらに推し進め、AIテンプレートのサンプルを紹介します。このサンプルは、機械学習を用いたサービスを商用展開するための一連の機械学習パイプライン処理を実装したものです。具体的には、MNISTデータセットを利用したOCRのユースケースの実装例を紹介します。

なお、本サンプルはLinux上のDocker CEで動作させています。執筆にあたり本サンプルの動作確認はCentOS上で行っています※1。

A.1 利用するデータセット

本サンプルで利用するデータセットは、MNISTの手書き数字データをもとにしたものです。0から9までの手書きの数字70,000点を収録したデータセットの中から、3個のデータをすでにサンプルのアーカイブ内の/src/main/resources/images/ディレクトリ配下に取り込んであるので、そちらを活用してください（サンプルのアーカイブの入手方法は後述）。特にMNISTのサイトからデータセットをダウンロードする必要はありません。

※1 CentOS用Docker CEについては、https://docs.docker.jp/engine/installation/linux/docker-ce/centos.htmlを参照してください。WindowsやmacOSで利用される場合には、Dockerfileで用いられるそれぞれのライブラリのパスを適切に変更する必要があります。また、Windows Home上でDocker Desktopを動作させるには、執筆時点ではWindows Subsystem for Linux 2（WSL 2）のインストールが必要です。詳細はhttps://docs.docker.com/docker-for-windows/install/を参考にしてください。



図A.1：入力データ（左から手書きの１２７）

ちなみに、全データセットを入手したい場合には、下記URLのMNISTのサイトからデータをダウンロードできます。

●MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges

```
http://yann.lecun.com/exdb/mnist/
```

A.2 サンプルの実行条件

本サンプルを含むアーカイブを入手するには、本書の紹介サイトにある「ダウンロード」の内容に従ってダウンロードしてください。本書の紹介サイトのURLは下記のとおりです。

```
https://book.impress.co.jp/books/1119101017
```

このアーカイブを入手したら、下記のようにツールやコマンドで展開してください。ここでは、現在の作業ディレクトリ配下にダウンロードして展開したものと想定しています。

```
$ unzip ai-template.zip
```

本サンプルの実行にあたっては、Dockerの環境を用います。Dockerを使う理由は、実行環境を簡単に用意できるためです。

ここからは、サンプルを実際に動作させる方法についてその概略を示します。Dockerのインストールも含めDockerの基本機能などの説明については本書のメインテーマから外れるため割愛します。Dockerに関する詳細情報が必要な場合には、Dockerに関するWeb情報や書籍を別途参照してください。

本サンプルでは、Dockerに実装されているDocker Composeという複数のコンテナのサービスを構築／実行する手順の自動化機能を用いて、アプリケーションを動かしていきます。

A.3 Docker 上でのサンプルの実行手順

以下では、Dockerの一般的な実行手順に沿ってアプリケーションを実行します。

1. Dockerfileの作成

このファイルでDockerコンテナの構成内容を記述します。本サンプルでは、すでにこのファイルの作成は済んでいるので、特に気にする必要はありません。このファイルには、Dockerの動作に必要な構成情報を記述しています。これにより、gitやmavenなどが環境にない場合に自動でインストールしたり、実行ファイルをコピーしたりするようになっています。

2. Dockerイメージの構築／起動

docker-composeコマンドを用いて、Dockerのイメージを構築／起動すると、アプリケーションが実行されます。このコマンドは以下のように実行してください。

```
$ docker-compose up -build
```

ちなみに、docker-composeを動かすには、事前にdocker-compose.ymlというファイルに起動の定義内容を記述しておく必要があります。この内容も今回のサンプルでは作成済みです。このサンプルの場合、mnist-templateのイメージをコンテナ名として指定して、ネットワークポートとしては8080を設定しています。docker-compose.ymlの一部の内容は、次のとおりです。

```
services:

  # ---------------------------

  # mnist-template

  # ---------------------------

  mnist-template:

    build: .

    image: mnist-template

    container_name: mnist-template

    ports:

      - 8080:8080
```

ここまでの手順により、mnist-templateのアプリケーションが起動されます。

3. アプリケーションの実行結果の確認

入力画像が7の手書き文字の場合、アプリケーションの実行結果は、以下のコマンドで確認できます。

```
$ curl -X POST -F "file=@src/main/resources/images/seven.
```

```
png" http://lloalhost:8080/mnist
```

このアプリケーションの出力結果は、以下のとおりです。この結果から7のクラスであるということを正しく予測できていることがわかります。

```
{ gclass h:7}    # <- result
```

以上が、サンプルの実行結果までを確認する手順です。これでパイプラインの各処理を簡単に設定して実行できることを確認することができました。

ここでは、機械学習パイプラインのデプロイツールとしてKonduit Serving（konduit-serving）を使用しました。このツールは、オープンソースとして公開されています。さらに詳しく知りたいという方は、以下のサイトの内容を確認してみてください。

https://serving.konduit.ai/

INDEX

た

な

著者

Adam Gibson（アダム・ギブソン）

Konduit 社の CTO。前職は Skymind 社の共同経営者。2010 年からオープンソースソフトウェアの開発を行っている。2012 年に Java ベースの機械学習システムである Deeplearning4j を発案。アメリカのミシガン州で育ち、数年間をシリコンバレーで過ごす。現在は東京在住。

著者／訳者

新郷 美紀（しんごう みき）

現在、AI 領域のソリューションアーキテクトとして活動中。東京工業大学卒、日本電気勤務。著書に『アプリケーションエンジニアのための Apache Spark 入門』（秀和システム）がある。また『詳説 Deep Learning —実務者のためのアプローチ』（オライリージャパン）の一部の翻訳を担当。2019 年と 2020 年に世界の人物年鑑 Who's Who in the World の Science and Engineering 部門に選出された。

STAFF LIST

カバーデザイン	岡田章志
本文デザイン	オガワヒロシ (VAriant Design)
本文デザイン・DTP	柏倉真理子
編集	大月宇美
	石橋克隆

■商品に関する問い合わせ先
インプレスブックスのお問い合わせフォームより入力してください。
https://book.impress.co.jp/info/
上記フォームがご利用頂けない場合のメールでの問い合わせ先
info@impress.co.jp

●本書の内容に関するご質問は、お問い合わせフォーム、メールまたは封書にて書名・ISBN・お名前・電話番号と該当するページや具体的な質問内容、お使いの動作環境などを明記のうえ、お問い合わせください。
●電話やFAX等でのご質問には対応しておりません。なお、本書の範囲を超える質問に関しましてはお答えできませんのでご了承ください。
●インプレスブックス（https://book.impress.co.jp/）では、本書を含めインプレスの出版物に関するサポート情報などを提供しておりますのでそちらもご覧ください。
●該当書籍の奥付に記載されている初版発行日から3年が経過した場合、もしくは該当書籍で紹介している製品やサービスについて提供会社によるサポートが終了した場合は、ご質問にお答えしかねる場合があります。

■落丁・乱丁本などの問い合わせ先
TEL 03-6837-5016 FAX 03-6837-5023
service@impress.co.jp
（受付時間／ 10:00-12:00、13:00-17:30 土日、祝祭日を除く）
●古書店で購入されたものについてはお取り替えできません。

■書店／販売店の窓口
株式会社インプレス 受注センター
TEL 048-449-8040
FAX 048-449-8041
株式会社インプレス 出版営業部
TEL 03-6837-4635

ディープラーニング構築（こうちく）テンプレート
[AI（エーアイ）プロジェクトの必須事項（ひっすじこう）と技術的指針（ぎじゅつてきししん）]

2020年9月21日 初版第1刷発行

著　者　Adam Gibson（アダム ギブソン）
著者／訳者　新郷 美紀（しんごう みき）
発行人　小川 亨
編集人　高橋隆志
発行所　株式会社インプレス
〒101-0051　東京都千代田区神田神保町一丁目105番地
ホームページ　https://book.impress.co.jp/

印刷所　音羽印刷株式会社
ISBN978-4-295-00986-3 C3055
Printed in Japan