AI 教育ガイドライン

2018 年 04 月 30 日

Contents:

第 1 章

イントロダクション

1.1 人工知能・機械学習・ディープラーニング

人工知能（AI）、**機械学習**、**ディープラーニング**といった言葉が飛び交い、どれも同じような位置づけに対する用語として用いられるケースが多い。しかし、厳密にはそれぞれに位置づけの違いが存在し、特に発注する際に、この用語の区別が曖昧であると受注者側への指示が不適切になる可能性があるため、注意が必要である。これら 3 つの用語に関する包含関係を以下の図 1 のような図で説明されることが多い。

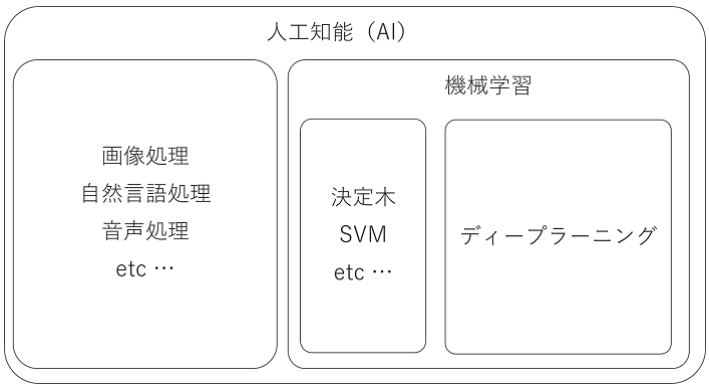
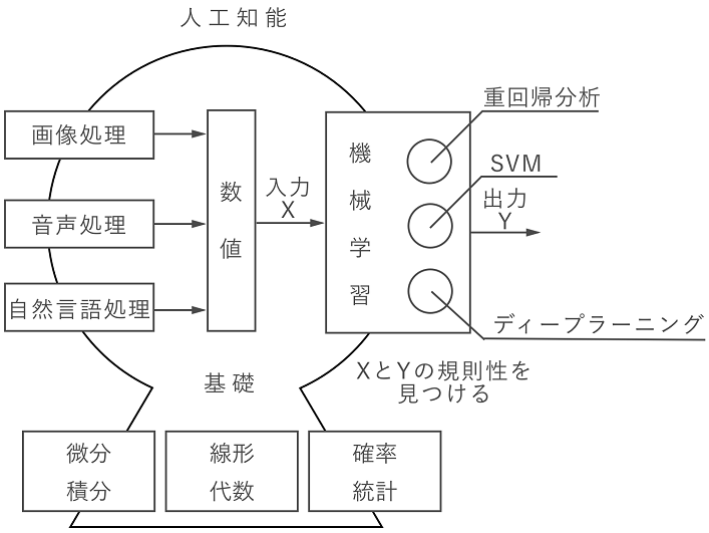


図 1: 人工知能・機械学習・ディープラーニングの位置づけ

こちらの図の通り、人工知能（AI; Artificial Intelligence）が一番の大枠である。そして、その AI の一部技術として機械学習が存在し、その機械学習の一部にディープラーニングが位置づけられるといった包含関係となっている。しかし、こちらの図で人工知能と機械学習の違いに関して理解できるだろうか。

そこで、もう少し具体的な事例を想像できるように図 2 を用いて、人工知能・機械学習・ディープラーニングについてのさらに詳細な説明を行う。

図 2: 人工知能・機械学習・ディープラーニングの具体的な処理の流れ



1.1.1 人工知能（AI; Artificial Intelligence）

人工知能は英語で Artificial Intelligence であり、その略語として AI が使用されるため、本稿でも今後は AI と呼ぶ。人工的に人間の機能を再現するもの全般に対して AI という用語が用いられている。人間のように考え、結論を出すものは広義に AI と呼ばれており、その中で機械学習のようにデータに基づいた高度な技術を使用したロジックでなく、人間があらかじめプログラミングを行ったルールベース通りに動くものも AI と呼ばれている。人間の機能を再現する方法としては、ルールベースや機械学習を用いた方法があるが、基本的には**学習**と**推論**によって成立しており、こちらの詳細は後述する。

機械学習では、入力データとして構造化された形式になっている必要があり、目や耳から入ってくる情報を処理するための分野があり、例えば画像処理や音声処理、自然言語処理である。データの構造化後は画像であっても自然言語であっても数値で表現されているため、コンピュータ内では画像や自然言語といった差がなくなる。

1.1.2 機械学習

機械学習では、入力データに対して、所望のアクション（出力）との関係性を見つける技術である。例えば、画像の情報が入力されて、この顔写真の画像は「佐藤さん」、この顔写真は「鈴木さん」といったように、紐付けを行うのである。この時の画像情報を入力、名前を出力と呼び、機械学習ではこの入力と出力の関係性を見つける概念である。厳密には入力データだけのケースもあるが、さらに詳細な説明は後述する。これで人工知能の一部として機械学習が位置づけられていることをおわかりいただけただろうか。

しかし、あくまで機械学習は入出力間の規則性を見つけるといった概念である。コンピュータに支持を出すためには、具体的な手順を明確に伝える手順書が必要である。そのような具体的な手順のことをアルゴリズムと呼ぶ。

1.1.3 ディープラーニング

ディープラーニングは機械学習アルゴリズムの一種である。入出力間の関係性のモデル化の仕方からモデル内に含まれているパラメータの調整方法までを具体的な手順として定義している。ディープラーニングが機械学習アルゴリズムの一種であるということは、別の機械学習アルゴリズムも存在しており、例えば、サポートベクターマシン（SVM）や決定木など多数存在する。最近ではディープラーニングがブームとなっているため、このアルゴリズムが注目を集める機会が多いが、現場ではディープラーニング以外の手法を用いられていることが多々あり、すべての問題に対してディープラーニングが万能でないことに注意が必要である。

このように、AI という大きな概念から、その問題を解くために必要なアクションに落としていった際に、機械学習やディープラーニングという技術が存在している。

1.1.4 基礎となるスキルセット

ディープラーニングを含めた機械学習を勉強するために必要なスキルセットは数学とプログラミングである。最近では、GUI (Graphical User Interface) による機械学習システムの開発も進んでいるため、プログラミングの知識が不要で解析を行うこともできるようになっている。実際には GUI でも組むことはできるが、解析では最初から良い結果が出ることは少なく、どのように改善を行っていくかの考察が重要になってくる。その際に、プログラミングや数学の知識がないと、この考察を行うことができない。そのため、最初の段階で試験的に組むことまでは数学やプログラミングの知識は不要で開発を行うこともできるが、長期的には必要になってくることが多い。

数学は基礎として、**微分積分**、**線形代数**、**確率統計**の３つを抑えておく必要がある。最初からすべてを手広く学ぶことは難しいため、まずはどのような位置づけでそれぞれが使われているかをイメージできる程度の理解で十分である。こちらの位置づけに関しては、後述している。

プログラミングでは、基礎的な文法を一通り勉強しておくと良い。大雑把にみると、プログラミングは繰り返しと条件分岐で成り立っているため、発注者側としては、フローチャートとしてグラフィカルに理解できる程度で十分である。実際にプログラムを組む際には、プログラミング言語を選定する必要があり、機械学習領域では Python というプログラミング言語を使用され、かつ非常に学びやすく直感的な言語であるため、初めての場合は Python でプログラミングを学ぶと良い。この領域の実案件でも Python を使用するケースが多いため、学び始めから本格的な導入までフォローすることができる。

1.2 学習と推論

AI の開発工程を理解する上で重要な工程の切り分けとして、**学習**と**推論**を区別しておく必要がある。機械学習をベースにした場合も、人間によるルールベースのプログラミングの場合も、学習と推論の工程は存在する。

1.2.1 学習

ユーザー視点では、「AI を使う」といった印象が多いが、実際には AI、特に機械学習の**モデルを学習させる**工程が一番最初に必要となる。

今回は例として、図 3 に示すように顔写真から名前を予測したいという問題設定で考えてみる。

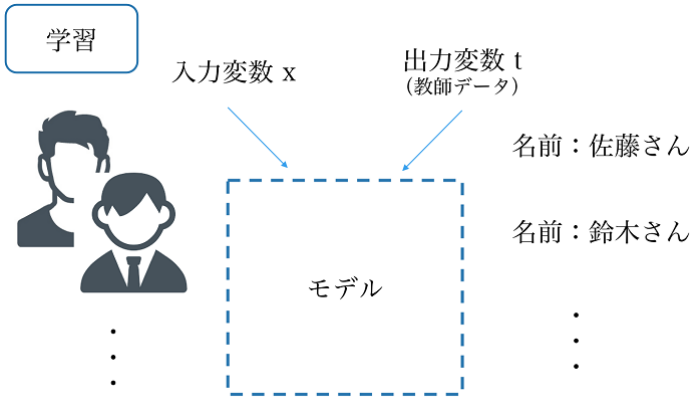


図 3: モデルを学習

当然ではあるが、機械学習においては、最初から顔写真をもとに名前が予測できるわけではなく、ある一定量のデータをもとに顔写真から名前を見分けられるような規則性を見つけ出す必要がある。このデータに基づいて規則性を見つけ出す工程のことを**学習**と呼ぶ。

学習の際には、入力とする画像に対して、それに紐づく名前と 1 セットで与える必要がある。もちろん最初は機械学習によって名前を予測することができないため、最初は顔写真に紐づく名前を手動でラベル付けする作業が必要となる。この時の人間側でラベルを付けて与える出力値を**教師データ**と呼び、t と表記されることが多い。そして、この入出力セットになった複数のデータを元に、機械学習のモデルを学習させる。モデルという言葉も概念的であるが、実態としては学習によって調整できる**パラメータ**と呼ばれる変数と入力変数の演算を取り決めた定式化されたものである。学習によって、うまく予測できるようなパラメータの調整を行う。「モデルの学習」という言葉が多用されるが、実態の処理としては、ある評価基準をもとにした「パラメータの調整」である。そして、学習によってパラメータが最適化され、その最適なパラメータを保持したモデルのことを**学習済みモデル**と呼ぶ。

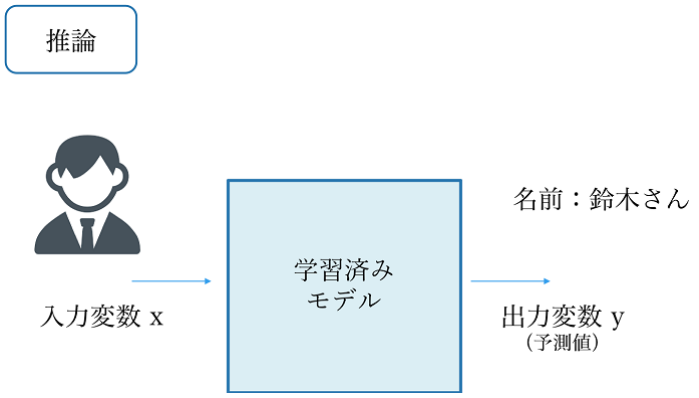
また、機械学習による学習の話に終始していたが、人間側で運用手順を明確に把握し、その手順をプログラムしていくことも一種の学習に含まれる。

1.2.2 推論

学習後に作成された学習済みモデルを使用して、図 4 のように、新しい入力に対する予測を行うことを**推論**と呼ぶ。

図 4: 学習済みモデルを用いた推論

AI を用いたサービスでユーザーが使用する際は、基本的にこの推論によって得られた値を使用している。推論によって計算される予測値は y で記述されることが多い。



ユーザー目線では学習の工程に気づかない場合があり、開発を行う際には、学習のステップを経て推論があることを忘れてはならない。

1.2.3 学習と推論のよくある誤解

ここで、実例を交えながら学習と推論の違いについてさらに理解を深めていく。

例えば、問題設定として、音声情報から文字情報を書き起こすことができる AI システムを作って欲しいという依頼がある。AI には詳しくないというクライアント側の主張によると、過去十数年分のデータが蓄積されていると聞いており、データの量は十分であると考えられる。データの質にもよるが、問題設定としては、機械学習を使えば実現可能であると一見考えられる。しかし、実際に案件が始まり、思いもよらない問題が発覚する。

クライアントから渡されたデータでは、音声情報が十数年分蓄積されているが、それに対応した文字の情報が全くない。今回の問題設定を振り返ってみると、入力は音声情報として出力は文字情報と設定する必要があった。しかし、学習を行うための入出力がセットになっていないため、文字を書き起こすようなモデルの学習を行うことができない。これが「データが存在する」という主張でよくある誤解である。

機械学習を適用するためには、アクションとして望ましい（今回の例では文字）情報と入力なる情報をセットで保存しておく必要がある。そのため、機械学習を使用することを将来的に視野に入れている場合、とりあえずデータを蓄積するのではいけない。入力となる情報は何か、出力として予測したい情報は何かを検討したうえで、その両方のデータをセットで蓄積しておく必要がある。

まとめとして、今回の問題設定である音声情報から文字情報を書き起こす機械学習モデルを構築する場合、次の手順で進めていく。まずは、音声情報と一緒に文字情報もセットで取得する。その際には音声ファイルの各時刻に対応する文字情報が必要であるため、単に音声ファイルとその文字が書き起こされたテキストファイルを保管しておけば良いわけではないため注意が必要である。データ蓄積のステップでは人間側の労力が大きくなるため、その点も考慮してスケジュールや予算を設定すべきである。そして、蓄積されたデータを用いてモデルを学習させる。うまく予測できるモデルが得られると、その学習済みモデルを使用してユーザー側で使用できる文字の書き起こし AI を提供できるようになる。

1.2.4 AI を現場へ適用する前に考えておくべきこと

システムを作る際に注意すべき点として、精度 100% の AI を作ることは現実的に困難であるということである。たとえ、大量のデータをもとにどれだけ学習させても、間違いが一切ないような AI を作ることが難しい。そこで、実際に使用される現場へ適用する手順を、企画の段階で詳細に詰めておくことが望ましい。具体的には、精度 100% が出ない際の運用でのカバーを考えておくと良い。

よくある AI 発注の間違いの間違いとして、最初から最後まで AI で解決するような仕組みを作ってしまうことがある。例えば、診断を AI が判定して処置を行うといったケースが考えられる。もし AI による診断の予測が間違ってしまった場合、間違った処置が施されてしまう問題が発生し、このリスクを考えると AI による診断を導入することはできない。

そこで、図 5 に示すような人間側での運用のカバーが重要になってくる。

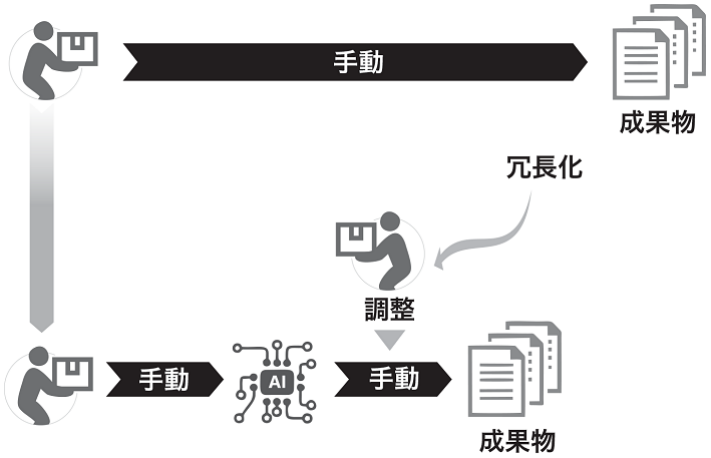


図 5: AI の間違いを運用でカバー

最初から最後まで AI に任せるわけではなく、まずは人間の手動運用で成果を確保できるように設計する。そして、その運用の中でデータを取得できて頻度の多い処理を AI によって学習させ、その部分を手動から自動へ置き換える。当然ではあるが、AI による予測精度は 100% でないため、得られた結果を人間が確認して、最終的な成果へとつなげていくフォローを行う。このように、最初から最後まで AI ではなく、運用フローの中で手動と自動を切り分けて効率化していくことが重要である。

診断の例では、AI によって得られた診断結果をもとに、医師の経験をもとに最終決定を下すフローが望ましい。AI によって推薦された診断結果によって抜け漏れの防止を高められることができ、最終的な判断も人間側で行うことができる。

1.3 AI が医療業界で使われている事例

医療業界において、特に目覚ましい発展は**画像**の解析である。2012 年以降、機械学習の一種であるディープラーニングに注目が集まって以来、画像領域への応用が進んできている。もちろん、ディープラーニングの登場より前から機械学習による画像の解析も行われていた。しかし、その際に障害になっていた点として、どのように画像を

解析するかが人間の経験と勘により行われており、専門知識が豊富かつプログラミングによる実装ができる医療知識のある優秀なエンジニアがいないとこの問題の解決は困難であった。そのため、エンジニアが医療分野の知識を身に着けるか、医師がエンジニアリング技術を身に着けるといった必要があった。そのため、実際に取り組むことができるのは、大学といった研究機関に限られていることが多くあった。それがここにきて、ディープラーニングという技術の登場により徐々に変わりつつある。

ディープラーニング登場以前と以後の違いについて、図 6 を用いて説明する

図 6: 従来の機械学習アルゴリズムとディープラーニングの違い

まずは機械学習を使用した物体の検出に関する流れである。画像処理の領域では、いきなり撮影などで取得した画像に対して物体検出などの解析をかけるのではない。ある対象とする現象（今回は顔）を特定しやすくするために、機械学習の入力データを適切な形に変換する前処理が基本的に必要である。今回の例では、顔を判断しやすいように、背景を削除したり、輪郭のみを残すような**特徴抽出**の処理を施している。そして、この特徴抽出済みの画像を入力データとして、機械学習による推論をかけることで、画像内のどの位置に顔が存在するかを判定することができる。この特徴抽出には長年の経験で培われた知識が不可欠であり、特定の領域に特化した特殊な処理であるほど、その経験を持ったエンジニアが少なく実装が困難であった。

想像していただきたい点として、ガンの発作位置を特定しやすいような画像の特徴抽出の方法をご存じだろうか。多くの方は知らないはずである。しかし、機械学習によって症状の発作位置を予測する場合には、このような専門的な知識が不可欠であった。

ディープラーニングが画期的であった部分は、この特徴抽出に関して理論上省略できることである。これまでの機械学習は入力、特徴抽出のロジック、教師データとなる出力の 3 つが必要であった。一方、ディープラーニングでは、この特徴抽出の仕組みをモデル内で表現できているため、入力データと教師データとなる出力の 2 つのみであれば良い。つまり、医師が病気の位置だけ特定したデータを準備しておけば、どのように病気に特徴抽出すると病気の位置を捉えやすいかといった特徴抽出のロジックを知らずとも、ディープラーニングが特徴抽出も含めた学習を行い、病気の位置を見つけ出せるようになる。もちろん、これは理論上の話であるため、実際の案件になるとデータを見て、ある程度の前処理は行うこともある。しかし、どのような特徴量が適切かの前提知識としてわかっていない業界に置いても適用可能となり、医療に関する専門知識を持たないエンジニアにおいても、医療現場のデータ解析が可能となってきている。

第２章

機械学習概論

2.1 機械学習の３大トピック

機械学習は大きく３つに分類され、**教師あり学習**、**教師なし学習**、そして**強化学習**である。開発する際、そして発注する際に、どの分野を選択すべきかを把握することが必要である。

2.1.1 教師あり学習

教師あり学習は、入力データに対して教師データがペアで揃っており、入力から出力を予測するモデルを作る。たとえば、「駅からの距離や部屋の広さ（入力）から家賃（出力）を予測する」や「画像（入力）から男性か女性といった性別（出力）を識別する」といった問題設定が挙げられる。

その中で教師あり学習では大きく分けて、**回帰**と**分類**があり、こちらもアルゴリズムを選択する際の大きな分かれ道となる。これらの違いは予測する値の形式である。回帰では予測値が**連続値**であり、分類では予測値が**カテゴリ**である。

回帰の例としては、家賃を予測する問題がわかりやすい。予測したい家賃は、50,000 円や 65,000 円といったように連続的な値の中から選択している。また、株価の予測や事故率の予測など、連続地で表現する場合はすべて回帰の問題設定である。回帰の有名な手法としては、**重回帰分析**や**ロジスティック回帰**、**ニューラルネットワーク**がある。

分類の例としては、赤ワイン・白ワインを当てるといった問題がわかりやすい。これは２つのカテゴリの中からどちらが近いかを考えるため、２クラス分類である。また、赤ワイン・白ワインだけでなく、ロゼも含めた分類を行いたい場合は３クラス分類となる。分類ではあらかじめ分けられているカテゴリに対して、どのカテゴリに一番属しているかを判定する。分類の有名な手法としては、**決定木**や**ニューラルネットワーク**がある。

2.1.2 教師なし学習

教師なし学習は、入力となるデータのみで、データの背後に存在する本質的な構造を抽出する。たとえば、「顧客データを似ている属性ごとに分ける」や「大量のデータの中から情報として密度の高い部分のみ抽出する」といった問題設定が挙げられる。

その中で教師なし学習では大きく分けて**クラスタリング**と**次元削減**が有名である。

クラスタリングでは、データ間の距離という概念を使って、近いもの同士で分けるといったことが行われる。これは一件、教師あり学習の分類に似ているが、異なるものであることに注意が必要である。大きな違いは、教師データが与えられているかである。分類は教師あり学習であるため、どのカテゴリに属するかという教師データがあらかじめ与えられている一方、クラスタリングは教師データが与えられていないため、真に属するカテゴリはわかっていない。そのため、クラスタリングにおいて、基本的には人間の事前知識と照らし合わせながら現場で活用していくことが多い。クラスタリングの有名な手法としては、**k-近傍法**などがある。

また、**次元削減**では、その名の通りデータの次元を削減するが、具体的にはデータのカラム数が多すぎる場合、例えば 5000 カラム程度ある場合に、これを密度の高いデータに変換する。次元削減後には 5000 カラムが 100 カラムほどの密度の高いデータに削減されるのである。データのカラム数が大きすぎると、計算することが困難になるケースに効果的である。次元削減を行うことで、情報量は損なわないように次元を削減して小さくすることができ、計算速度を向上させることができる。特に、EC 領域などユーザー数やアイテム数が膨大なケースにおいては欠かせない技術である。計算速度の向上以外に、入力となるデータの変数同士で相関を持つ場合において、予測に悪影響を与えるようなケースが知られており、これは**多重共線性**と呼ばれる問題である。そのため、入力変数同士で相関を持たないように無相関な変数に変換（次元削減）したのちに、その変数を使って機械学習のモデル構築を行うことで、この多重共線性の問題を回避することができる。このように、回帰や分類といった教師あり学習の前処理として使用される機会も多い。次元削減では有名な手法には」、**主成分分析**がある。

2.1.3 強化学習

強化学習はある環境内におけるエージェントが、現在の状態を観測し、取るべき行動を決定する。一言で説明するなら身近なお掃除ロボットを想像していただきたい。実際には全く始めデータがない状態の中、つまり部屋の床に置かれているかもわからず、かつ部屋の広さや家具の配置情報も全くわからない状態の中でスタートする。そして、動きながら部屋の情報を収集し、そのデータに基づいて、部屋を最短で掃除できるように行動を決定していく。

強化学習に関しては、AlphaGo と呼ばれる AI が囲碁において人間に勝利したときに脚光を浴びた。2 つのモデルを競わせることでより強い AI へと強化を続けていったのである。

強化学習はロボットとゲームの領域で大きな成果を上げている。理論上、強化学習では、自動的に動きまわったり対戦を繰り返すことで強くなってため、実社会での幅広い応用が期待されるが、逆になぜロボットとゲームの領域でしか大きな成果が挙げられていないのだろうか。発注の仕方として、あとは強化学習でどんどん強化させれば、最初のデータがなくても自動的にデータを収集しながら AI が育っていくという夢物語に近い話もでることがあるが、なぜそれはロボットとゲームの領域が多いのだろか。

その適用できるかを分ける点はシミュレーションを行うことができるかである。ロボットでは物理法則に基づいた

シミュレータにより実環境における振る舞いにある程度近い動きをシミュレータ内で再現することができる。ゲームでは、ルールという絶対的な法則をもとに動くため、当然再現をすることができる。そのため、そのシミュレータをベースに何百回、何千回、何万回とうまく動くための動きを習得できるまで何度も何度も失敗を繰り返すことができる。当然、AlphaGo も最初は人間でも楽に勝てる程度であるが、これをシミュレータに基づいて何度も学習させることで、人間では到底かなわない動きを身に着けることができるのである。まとめとして、強化学習は最初からデータが少ないもしくはない状況から始めることができる柔軟性の高い手法であるが、一方シミュレータのように何度も繰り返せる環境が必要となる。もちろん、実機を準備して何度も動かして学習させることができれば、強化学習に基づいてその行動を徐々に強化していくことも可能である。

2.1.4 教師あり学習・教師なし学習・強化学習の違い

機械学習における３つの大きなトピックを紹介したが、実問題に適用していく際にはどのような基準で選択すれば良いかを明確にする。

まずは、教師データの有無である。教師データが存在する場合、教師あり学習を使用することになる。この際には、入力となるデータは何か、そして教師データは何かを明確化すれば良い。そして、その教師データが連続値であれば回帰、カテゴリであれば分類の手法から選択すれば良い。さらに、入力データのカラム数が大きすぎて計算が困難になるようなケースは教師なし学習である次元削減を使用したのちに、教師あり学習を適用することも検討できる。

次に、教師データがない場合である。教師なし学習になり、多くの問題はクラスタリングである。クラスタリングでは、あらかじめクラスターと呼ばれる属する集団の数を人間側で決めておく必要があるため、どの程度のクラスターの数が潜在的に存在するかをそのデータに関する知見に基づいて定めることがある。また、データの特性を可視化したい際に２次元もしくは３次元に次元削減することもしばしば用いられる。これによりデータの特性を把握することに繋がるケースもあるため、覚えておくと良い。

そして、強化学習はシミュレータを含めた何度も実験できる環境を準備できるかがである。

2.1.5 ビジネス活用を視野に入れた手法の選択

それでは、ビジネス活用を視野に入れた際によく使用するものは、教師あり学習、教師なし学習、強化学習のどれが多いか。ビジネスの現場において、効果を発揮できるのは教師あり学習が多い。もちろん、教師なし学習や強化学習に関しても、研究開発で使用されているケースは多々あるが、最終的な納品を考えると教師あり学習でないと難しいことがわかる。開発者が納品する際には最終的に予測精度が何%程度確保できているかといった結果を把握したうえで納品する必要があり、発注者として知っておく必要のある事項である。

それでは、教師なし学習のケースから考えていく。例えば、教師なし学習には異常検知で有名な 3σ 法と呼ばれる手法がある。過去のデータが平均 μ、標準偏差 σ の時、新たなデータが統計的には $\mu \pm 3\sigma$ に入っている確率が 99.7% である。 そのため、この $\mu \pm 3\sigma$ の領域に入っていなければ異常、入っていれば正常といった判定基準で異常検知を行うことができる。この手法では、異常か正常かといった教師データはなく、入力となるデータのみで異常の定義を行っており、教師なし学習である。つまり、異常なデータがまったくない場合においても異常と正常の

定義を行うことができるのである。医療の領域では、健常者の方が圧倒的に多く、発作の症状といった異常の際のデータを集めることは極めて困難である領域では、正常のみでモデル化をできる恩恵は大きくある。

しかし、モデル化はできたものの、納品の際に問題が発覚する。過去の異常データがないもしくはほとんどないため、実際に今回のモデルを現場に導入した際にどの程度うまく検出できるかを検証できないのである。そのため、モデルは作ったものの、結局何パーセントぐらいの精度が出るかは、現場に投入してみるまで分からない。そうなると実際にどのぐらいの精度が出るかわからないものはリスクが高く、導入するための意思決定ができないことになる。このようにデータを解析してもっともらしい結果を出すことは可能であるが、現場へ導入する際の意思決定ができないケースがあることまで考慮して、手法の選択を行う必要がある。

教師なし学習に対して、教師あり学習では、手持ちのデータを学習に使用する**訓練データ**と検証のみで使用する**検証データ**に分割する。検証データもあくまで過去のデータではあるが、学習には使用していないデータとして残しておくことで、未知のデータを模して検証を行うことができる。例としては、受験勉強で例えるとわかりやすい。過去 10 年分の過去問を購入して、すべての過去問で勉強をしてしまうと実力の確認ができなくなってしまうため、前半の 5 年を勉強用（訓練データ）に使用し、後半の 5 年を実力テスト用（検証データ）として使用するのと同じである。この例にもあるように、もちろん検証データも過去のデータではあるため、未来に対する結果を保証するものではない。しかし、この結果があるとないでは導入に対する根拠が大きく違ってくる。

そのため、発注者側としては、この意思決定まで考慮したうえで発注することが望ましい。また、手持ちのデータを渡して訓練データと検証データを開発側で適当に分割してしまうと、恣意的に予測精度を操作できてしまう。そこで、手持ちの 10 年分のデータに対して、最新 1 年分を検証データとして、残りを訓練データとして分割して進めるといった方針の決定も行っておくことが望ましい。

2.2 機械学習アルゴリズム

実案件に取り組んでいく際の選定基準が把握できたうえで、次に具体的にどの機械学習アルゴリズムを使用して解析を行っていくかを決定する。開発者側に任せるという選択もできるが、どのような手法がどのような特性を持っているかの概要を把握しておくことが望ましい。そこで、本節では、有名な機械学習アルゴリズムについて、具体的なアルゴリズムの話は省き、発注の際に重要である、どのような特性でどのような事例に適しているかを主に解説する。各手法の注意点に関しても補足を加えており、まずは用語としてどのようなものがあるかを抑えておく程度で十分である。

2.2.1 教師あり学習

重回帰分析

重回帰分析は Excel 等でも簡単に実装を行うことができる代表的な手法であり、ひとまず基準として試しておくと良い手法である。回帰向けの手法であるため、連続値を予測値とする。

入力変数が M 個の場合において、出力 y の予測値が $y = w_1x_1 + w_2x_2 + \ldots + w_Mx_M$ のように計算される。 ここで、$w_1, w_2, \ldots, w_M$ はパラメータと呼ばれ、データに基づいて調整される変数である。このように、パラメータ×入力を足し合わせた構造を線形結合と呼び、このような回帰を**線形回帰**と呼び、重回帰分析は線形回帰モデル

である。重回帰分析の特徴として、パラメータ×入力を変数の数だけ足し合わせたシンプルな構造となっているため、どの変数が予測に寄与しているかなど、解析した結果の説明を行うことができるその一方、複雑な構造のデータにフィットさせることが難しい場合があるため、注意が必要である。先述した入力変数間に相関を持つ場合に生じる多重共線性の影響を受ける手法であるため、入力変数間の相関の確認も重要である。

ロジスティック回帰

ロジスティック回帰は医療の領域でも昔からよく使用されてきた回帰手法である。重回帰分析の線形回帰モデルとは異なり、ロジスティック回帰は**非線形回帰**であり、**一般化線形モデル**とも呼ばれる。回帰という名前がついているが、分類の問題に使われるケースが多く、あるカテゴリに属する確率を予測する。つまり、ロジスティック回帰により予測した確率の値が最も大きなカテゴリに分類するのである。計算量も少ないため、分類を実装する際には、基準として試しておく手法である。

Support Vector Machine

回帰と分類の両方を兼ね備えた非線形な手法である。**カーネルトリック**と呼ばれる技法を使うことで、とても柔軟にモデル化を行うことができる。また、計算量も比較的少ないため、現場でよく用いられている手法である。

ハイパーパラメータと呼ばれる手法の中で必要な値をあらかじめうまく設定する必要がある。この設定がうまくいっていない場合、精度が低かったり、訓練データでは精度が高いが検証データでは精度が低い**オーバーフィッティング（過学習）**という現象が起きるため、注意が必要である。この場合、**ハイパーパラメータ**を適切に調整することで回避することができる場合がある。このハイパーパラメータの調整には**クロスバリデーション**と呼ばれる手法を用いることが多い。

決定木

古典的な機械学習アルゴリズムの一種であり、現場で使われることが多い手法の一つである。回帰と分類に対応しており、非線形な手法である。人間の意思決定のように木構造に分けて考えていくため、最終的にどの変数が寄与していたかを定量的に議論しやすい。また、ほかの手法は変数同士のスケールが異なる際に学習への悪影響を受けることが多いが、決定木はその影響を受けにくい。

複数の決定木を並列につないでノイズの影響を受けにくく考慮した**アンサンブル学習**である**ランダムフォレスト**も決定木の一種である。ランダムフォレストをはじめとした乱数を使用する手法では、**乱数のシード**と呼ばれる値を固定しておかないと**再現性の確保**ができないため、その点も検証時にチェックしておく必要がある。

ガウス過程

ガウス過程は基本的に非線形な回帰手法である。ロジスティック回帰同様、確率の値を予測することで分類の問題設定にも対応可能である。**ベイズ統計**がこのアルゴリズムの基礎になっている。

ガウス過程の大きな特徴としては、出力の値を予測するだけでなく、その出力の値に関する標準偏差も計算される。つまり、その予測値に関して、統計的にどの程度不確かさがあるかを定量評価することができる。これはほか

の手法にはない特徴である。この特性を利用して、効率的に実験していく**ベイズ最適化**というアルゴリズムも提案されている。

ディープラーニング（ニューラルネットワーク）

すでに登場しているであるが、脳のニューロンをもしたネットワーク構造でモデル化した非線形な手法であり、回帰と分類の両方に対応している。**特徴抽出**を行う構造がモデルの内部で考慮されており、画像や自然言語といった特徴抽出が難しい領域においても柔軟に対応することができることが大きな特徴である。ほかの手法と比べると計算量が多く、GPU (Graphic Processing Unit) による演算の高速化が不可欠であり、計算機環境の準備において、ほかの手法に比べて敷居が高いことに注意が必要である。計算量は多いが、逆にほかの手法に比べると数学的な敷居は比較的低い。

画像向けの CNN (Convolutional Neural Network) や時系列向けの RNN (Recurrent Neural Network) などディープラーニングの中で、それぞれのデータの特性に合わせたアルゴリズムが提案されている。

2.2.2 教師なし学習

K-means 法

クラスターの中心（各クラスターに属するサンプルの座標の平均）を用いて、どのクラスターに一番近いかを判定する手法である。使用する際の特徴としては、学習の前にクラスターの数をハイパーパラメータとして指定する必要があるため、使用するデータセットに何クラスター程度存在するかを把握しておかないとうまく使用することができない。

また、教師あり学習の分類とは異なり、あるクラスターへ分けることができたとしても、それが何を意味しているのかは人間側で別に考察が必要となるため、とりあえずクラスタリングという使い方では得られるものが少ない。すでにある知見に基づいた仮説と、その立証のためにクラスタリングを使用することが望ましい。

主成分分析

入力変数を別の**潜在変数**と呼ばれる新たな変数へ変換することで、データのカラム数を圧縮する手法である。データの特性を可視化して考察するために用いられるケースも多い。

使用する際の注意点として、最終的な潜在変数の数をハイパーパラメータとして指定する必要があり、この値を決めることが難しい。教師あり学習の前処理として使用する場合は、教師あり学習の予測精度が向上するような潜在変数の数とすれば良いため決定しやすい。

2.2.3 強化学習

Q-学習

強化学習では、現在の**状態**における次の**行動**を決定していき、その際の行動指針は**報酬**を多く獲得できる方向へ進むことである。厳密には目の前の報酬ではなく、将来的に得られる報酬を最大化できるような方向を選択する。しかし、将来的な報酬を最大化するように目の前の行動を判断することは当然難しい。そこで、Q 学習では、各選択肢に対して、**Q 値**と呼ばれる値を将来的に得られると推測される値を割り振り、各行動によってその Q 値を更新しながら学習していく。

2.3 演習課題

2.3.1 問題設定

実際に機械学習の位置付けやアルゴリズムに関して概要を把握できたところで、演習問題を行う。今回は MRI の画像データから、腫瘍位置を機械学習アルゴリズムによって発見する機械学習モデルの構築を依頼する設定である。

手元にあるデータとしては、図 7 に示すような脳画像が数百〜数万枚あるとする。

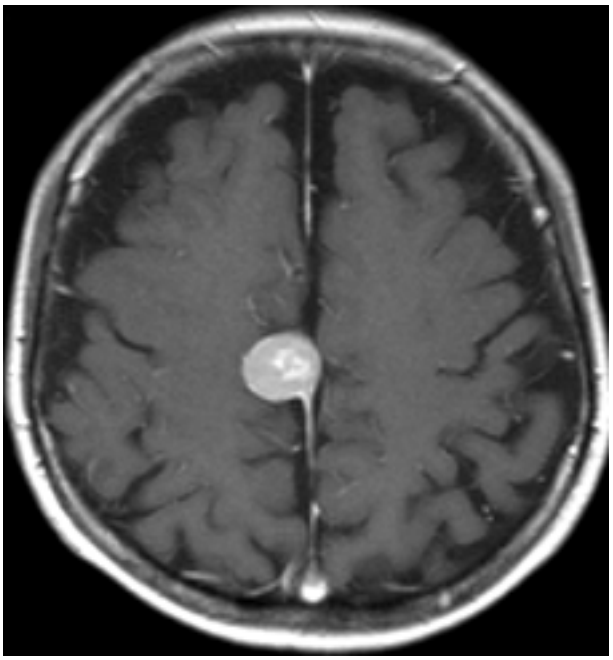


図 7: 入力データとなる脳画像

さて、ここで問題である。発注者の立場では、どのような機械学習アルゴリズムを使うかは考えなくても良いとするが、教師あり学習を選択するとなると、入力データとなる画像以外に教師データの準備を行わなければいけない。依頼する際には開発者側は医師ではない可能性が高いため、教師データのラベル付けは医師側で行っておくこ

とが望ましい。

それでは、機械学習を適用する際に、どのような教師データを準備しておくとスムーズに機械学習アルゴリズムのモデル構築を依頼できるかを答えなさい。作成すべき教師データのヒントは、運用に入る際に画像を入力としてどのような情報が得られると望ましいかである。

2.3.2 解答

教師データとして、どのような形式のデータを準備しておくと望ましいかを考える際のヒントとして、運用に入る際に画像を入力としてどのような情報が得られると望ましいかであった。実際に診断で使うことを想定すると、画像を入力情報として「腫瘍がどこにあるのか」を予測させることが望ましい。さらに具体的に落とし込んでいくと、「どこに」「腫瘍がある」ことを見つけたいということである。これは 2 つの手法の組み合わせである。

まず 1 つ目が腫瘍か否かを問わず、物体を検出する手法であり、「どこに」に対する解答となる。このどこにを数値で表現すると、**座標情報**が必要である。そのため、図 8 に示すように、腫瘍であると判断した個所の座標情報を教師データとして抽出する。

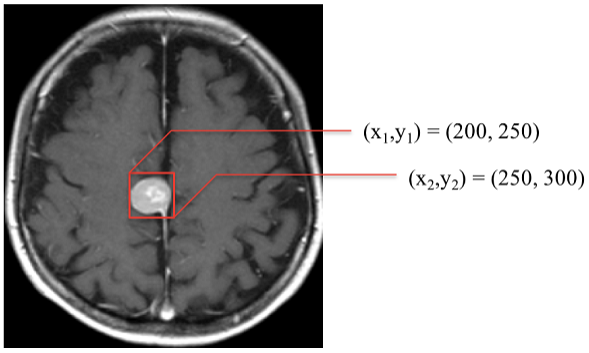


図 8: 教師データのラベル付け

そして、もう一点、「腫瘍がある」に対応したラベル付けも必要である。検出された座標の物体が「腫瘍」であるのか、それとも「骨」か「血管」かなど、物体にも複数の種類が考えられる。そこで、この指定した座標は「腫瘍」であるといったカテゴリの指定も必要となる。今回の例では、検出する物体が腫瘍の一種類であるため、このカテゴリに割り振りは省略することもできる。

まとめとして、各画像に対して、教師データには「座標情報」と「カテゴリ」の 2 点に関するラベル付けを行うことができていれば良い。このように教師データには最終的に運用の際に欲しい情報を取得しておけば、依頼の際にもスムーズに進めることができる。

第 3 章

機械学習の周辺技術

前章までは機械学習の大枠の概念や具体的なアルゴリズムについて解説した。本章では、その機械学習の実装に必要な周辺技術に焦点を当てて解説する。

3.1 機械学習に必要な数学

先述したとおり、機械学習に必要な数学は**微分積分**、**線形代数**、**確率統計**の大きくわけて 3 つがある。この 3 つが機械学習を使用していく上で必要不可欠といえる。しかし、最初から全てを習得することは難しく、特に発注側の立場であれば、議論をする際に抑えておくべき用語とそれらの数学が使用される用途だけ把握しておけば最初のステップとしては良い。

3.1.1 微分積分

まず微分積分であるが、高校の時は微分と積分をセットで習うことに対し、機械学習では微分と積分は別々の用途に使用される。特に最初の段階では微分に関する概要とその働きを抑えておくことが重要となる。

まず微分は「何が求まるか」を整理しておく。微分では**接線の傾き**を求めることができると習っている。

それでは、接線の傾きが求まると「何に使えるか」が重要であり、図 9 に示す例をもとに解説を行う。

この例では、$y = x^2$ のグラフであり、その微分して求める導関数は高校の時に習った公式を思い出すと、$y' = 2x$ である。

さて、この導関数は何に使えるのであろうか。オレンジの線は $x = 3$ の時における接線であり、緑の線は $x = -2$ の時の導関数により求まった接線である。この接線の傾きがこの導関数によって求まり、オレンジの線の場合、$x = 3$ のため、$y' = 2 \times 3 = 6$ となり、傾きが正のときは右肩上がりの直線となる。緑の線の場合、$x = -2$ のため、$y' = 2 \times (-2) = -4$ となり、傾きが負の時は右肩下がりの直線となる。 そのため、図 9 のようにこの接線の傾きを使用して接線を描くことができるが、何に使えるのかが不明なままである。

それでは、赤い線の $x = 0$ で考えてみる。たとえば、この $y = x^2$ の関数が例えば、機械学習における実際の値と

予測値との誤差を表しているとする。誤差を一番小さくできるようなパラメータに調整することで、最も良い予測を行う学習済みモデルを得ることができる。つまり、誤差が最小となるパラメータの値を求めたいというモチベーションが機械学習にはある。そこで、今回の関数において最も小さくなる点は $x = 0$ の点であり、この点をアルゴリズムとして探索する方法を考えると、傾きが右肩上がりでも右肩下がりでもなく、水平になるような点が最小となる点であることがわかる。つまり、接線の傾きの値が 0 となる点を見つけることで、誤差が最小になる点を見つけることができるのである。そのために、微分が使われる。補足として、厳密には、関数の形を考慮して先述の議論を進める必要があるが、今回は大枠としての理解を優先するためにその議論を省略している。

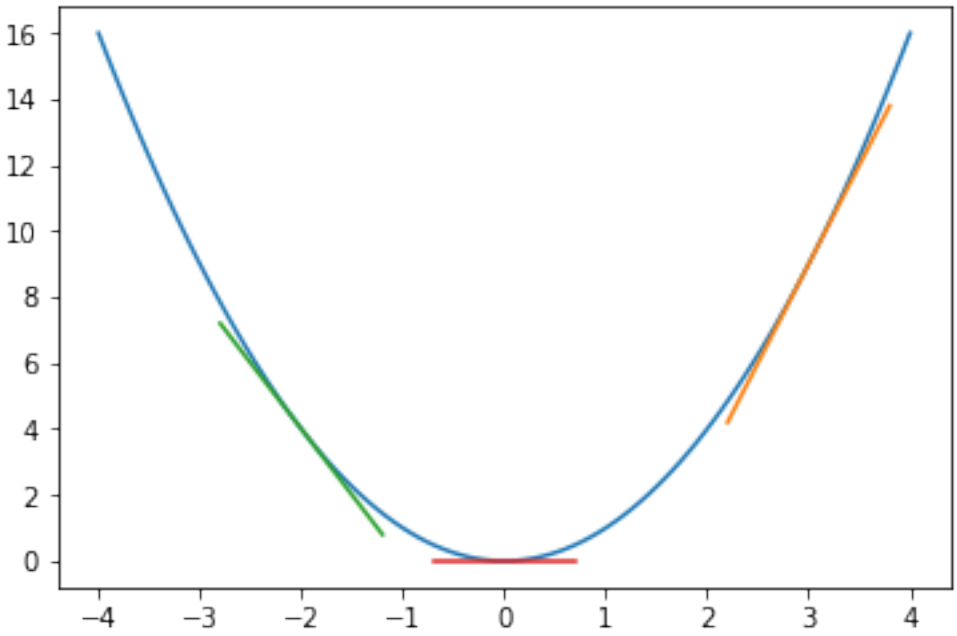


図 9: $y = x^2$ とその接線

積分は**ベイズ統計**と呼ばれる数学的に難易度が高い機械学習の手法で登場し、主に時系列データなどのある区間に関する**面積**を計算するということに使用される。

3.1.2 線形代数

機械学習を勉強する際に最も重要だとよく言われるのが**線形代数**である。線形代数は難しい学問であるかというとそういうわけではない。機械学習での議論はほぼすべて線形代数の知識を前提としているため、線形代数を学んでおかないと議論についていくことができないのである。

まず確実に覚えておかないといけない用語として、**スカラー**、**ベクトル**、**行列**がある。この 3 つの明確な違いを説明できるだろうか。ベクトルや行列といった言葉は開発するシステムの仕様をすり合わせる議論の際にも頻出するため、この用語を正確に把握しておかないと企画段階で理解できなくなってしまうのである。

スカラーは x や y といったひとつの変数を指す用語である。

ベクトルはそのスカラーをまとめたもので、

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

のように記述する。

行列はさらにそのベクトルをまとめたもので、

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

のように記述する。

このようにスカラー、ベクトル、行列は特別な知識ではなく、変数をひとまとまりにして記述して扱うことができるための定義であるため、特に難しいものではない。機械学習では、データのサンプル数も入力変数の数も複数であるため、スカラーだけでは表現しきれず、これらの情報をまとめてベクトルや行列という形にすることで、非常にスッキリとした議論を行うことができるのである。

そして、このベクトルや行列に関して、少ないコストで効率的に計算する方法を知ることが線形代数を学ぶ大きな目的である。特にわざわざ難しい計算をしようとするわけではなく、情報をうまく整理して扱うための学問として線形代数が用いられている。

3.1.3 確率統計

機械学習といえばまず確率統計が必要という印象が強いかもしれないが、意外にも初歩的な機械学習では確率統計の知識がそれほど大きくは必要ない。

線形代数をベースとした機械学習では、**平均**、**分散**、**標準偏差**、そして**正規分布**といった初歩的な確率統計の基礎を抑えておくだけで議論に参加できる。

そして、**ベイズ統計**と言われるさらに難しいな手法を勉強する際には、**確率密度関数**であったり、そこから派生していく難しい数式が必要不可欠となる。こちらは、開発側が把握しておけば良いため、発注者側としてはそれほどの深掘りは一旦必要ない。

それでは、最低限の議論に必要な平均、分散、標準偏差の計算を演習として行う。

$$\boldsymbol{x} = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

に関する平均 μ、分散 σ^2、標準偏差 σ を求めよ。

平均 μ

$$\mu = \frac{1}{5} \times (-1 - 2 + 0 + 1 + 2) = 0$$

分散 σ^2

$$\sigma^2 = \frac{1}{5} \times \{(-2-0)^2 + (-1-0)^2 + (0-0)^2 + (1-0)^2 + (2-0)^2\} = 2$$

標準偏差 σ

$$\sigma = \sqrt{\sigma^2} = \sqrt{2}$$

平均は全体の中心を表しており、標準偏差と分散は中心からのデータのばらつきを定量評価できる指標である。

3.2 プログラミングと環境開発

機械学習は数学だけではなく、**プログラミング**の知識も必要となる。また、プログラミングを行うに伴って、**OS**、**プログラミング言語**、**フレームワーク（ライブラリ）**、**API** といったソフトウェアに関するその周辺知識も必要となる。

機械学習のシステムを含んだ Web アプリケーションを開発する際に、それぞれ使われている技術や用語に関して、その流れも含めて図 10 に示す。

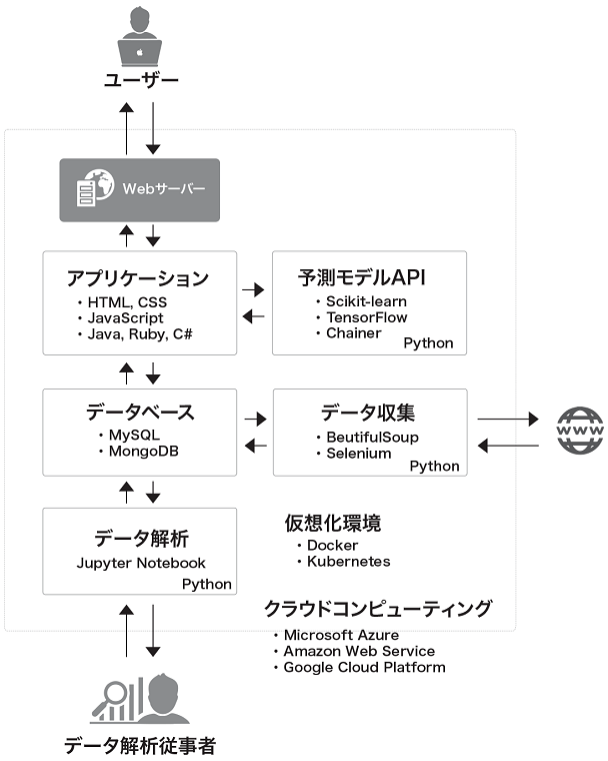


図 10: Web アプリケーション内で使われている技術や用語

こちらの画像からわかるように、実際の開発を依頼するにしても、これらの仕様や要件を設計段階で指示できる必要があるため、どのような役割であるかを明確に理解しておかないと危険である。

最初からすべてを深く学ぶ必要はないが、それぞれ出てくる固有名詞が大枠でどの位置づけにあるものかは常に理解しておく必要がある。そのため、まずプログラミングに必要な大枠の概念について、それぞれ紹介していく。

3.2.1 OS（Operating System）

OS とは、その概念の説明よりも、Windows や OS X、Linux といった具体的な名称を聞くとわかりやすいが、ハードウェアの制御とユーザーが使用するアプリケーションのベースを取り持つ役割の部分である。基本的には、Windows や OS X を搭載した PC で解析を行う機会が多いが、ディープラーニングなど大量の計算が必要なアルゴリズムを使用する場合には、手元の PC（ローカル）ではない、インターネット越しに存在するサーバー（リモート）を使用して計算することもある。このリモートにあるサーバーでは Linux を搭載しているケースが多く、特に機械学習系では Linux ディストリビューションのひとつである Ubuntu を使用する機会が多い。そのため、ローカルの環境が Windows や OS X であるからと Linux に関する知識が不要ではない。

図 11: ローカルとリモートの計算機環境

また、OS ごとに解析環境の設定が大きく異なることも初心者への大きな障害となる。Windows で使用していた GUI での環境構築が Linux で通用することはほとんどなく、基本的に CLI (Command Line Interface) によるコマンド操作でインストールからプログラムの実行まで管理する必要がある。さらに、Linux 用の環境構築手順も別途調べなおす必要があり、大変手間のかかる作業であるため、開発を本格化させるのでなければ最初は避けて通るほうが無難である。

最近では、**Docker** や **Kubernetes** と呼ばれる仮想化技術の登場により、OS 間の差を埋めることができるようになっている。これらの技術を使えば、ローカルで構築していた環境を、リモートへそのまま移植できるようになっており、ローカルとリモートによる差の障害は少なくなっている。ただし、これらの仮想化技術、たとえば Docker では、Windows に Docker をインストールし、その Windows 上の Docker で環境構築等の操作を行うが、この時の Docker 上の環境は Linux である。そのため、ローカルの Windows 上でも Linux で操作しておき、リモートの Linux 上でも Linux で操作するだけであるため、これらの仮想化技術を使えば、Linux に関する知識の習得を省略できるわけではないので、注意が必要である。

こういった仮想化技術で共通で使用されているため、Linux 特に、Ubuntu をベースとした開発に慣れておくことを推奨する。

3.2.2 プログラミング言語

コンピュータへのプログラミングを行う際には、プログラミング言語をひとつ選択して進める。プログラミング言語には、**C 言語**や **Java**、**Ruby** や **Perl** など様々なプログラミング言語があるが、機械学習に関する解析およびそのシステムを作成する場合は、どの言語が最も適しているのだろうか。

一般的に、機械学習でよく使用されるプログラミング言語は **Python** である。 それ以外にも **R 言語**や **MATLAB** といったデータ解析向けのプログラミング言語も存在するため、そちらを使用しても構わない。現在は、多くの解析案件に Python が使用されていると言われており、特に問題がなければ Python を発注する際のプログラミング言語として使用すると良い。

Python が好まれる理由としては、もともと Web アプリケーション向けの言語として開発されているため、単なるデータ解析だけでなく、出来上がった学習済みモデルをシステム統合すると際にもシームレスに実装を行うことができる。また、Python を使用した機械学習のコミュニティが非常に盛り上がっており、その結果 Python のプログラミングに関するリファレンスがインターネット上で多く存在する。これは、初心者でも勉強する敷居が低くなっていることを意味しており、これから採用する際の育成も容易であることにも繋がる。どのプログラミング言語を使用するかということは一番最初に決定する必要があり、最終的に**人の採用**であったりとか**引き継ぎの容易さ**といった観点まで考慮することが大事である。さらに、**Jupyter Notebook** という Web ブラウザベースで開発できる環境も準備されており、図 12 に示したリモートの環境を使用して開発する際にも、Web ブラウザをベースに開発することで、ローカルであってもリモートであっても差異を感じることなく開発することができる大きな利点がある。

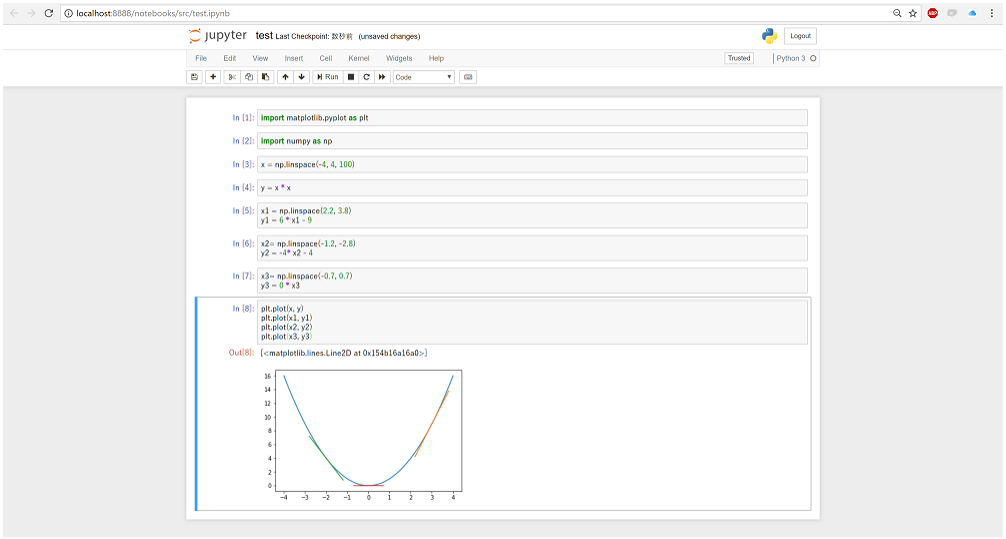


図 12: Jupyter Notebook による Web ブラウザベースでの Python の開発環境

3.2.3 フレームワーク

プログラミング言語を選択した次に選択が必要なものとしてフレームワークがある。今回は Python をプログラミング言語として選択した前提で話を進めることにする。

機械学習の実装を行う際には、すべてを一から実装する（フルスクラッチという）ことも可能であるが、現場での解析ではフルスクラッチで組むことは望ましくない。まず第一にフルスクラッチで組むことは時間がかかる。しかし、大きな問題は、解析の結果がうまくいかない際に、手法の選定を間違えているのか、プログラムの間違いであるのかといった問題の切り分けが複雑になることである。

それに対して、フレームワークでは、解析に必要な機能がすでに準備されており、その中で必要な処理をつなぎ合

わせていくだけで解析を実行できるようになっている。たとえば、機械学習では **Scikit-learn** と呼ばれるライブラリが代表的であり、このライブラリを使用することでプログラムの間違いによる可能性を大幅に下げることができ、解析の本質に集中できる。ディープラーニングのフレームワークも盛んに開発が進んでおり、Google 社製の **TensorFlow** や **Keras**、日本の Preferred Networks 社製の **Chainaer** など選択して使用すると良い。選択基準としては、リファレンスの多さや周りの知人が使っているものを選択すると良い。

また、機械学習のフレームワークだけでなく、最終的に Web のシステムへ統合していく際には、図 10 に示すように学習済みモデルを配置した推論サーバーを作成する必要がある。その際には、Web フレームワークも使う必要があり、Python では、**Django** や **Flask** などが有名である。推論サーバーを作成するためには、Web アプリケーションに関する知識も必要となってくるため、開発者サイドにデータ解析と Web アプリケーション開発ができる人がいるかの確認を行っておくと推論のステップで滞りなく進めることができる。

3.3 機械学習がよく使われる領域

3.3.1 画像

近年、画像データの解析に関して非常に盛り上がっている。画像データは単純に入力変数として使用する際には縦かける横、さらにカラー画像の場合は CHANNEL という部分も追加されて、非常に多くの画素数になる。例えば、縦が 100 ピクセル、横が 300 ピクセルのカラー画像の場合、100 かける 300 かける 3 チャンネルで合計 9 万ピクセルとなる。単純な機械学習の入力変数として扱う場合入力変数の数が 90000 となる。実際には 1 枚の画像であるため 90000 の入力変数という問題設定にするとモデルの学習が非常に難しくなってしまう。

ここでパラメータの数を減らすということが非常に大きな問題である、この解決策として、**畳み込みニューラルネットワーク（CNN）**が登場している。これは古くからあった画像処理のフィルタという概念をパラメータとして適用することでパラメータの数を減らし、かつ画像としての特性もうまく考慮しながら学習させることが可能になっている。CNN はディープラーニングの一種であり、医療画像に関してもこの CNN によって多くのブレークスルーが起こっている。

3.3.2 時系列

時系列に関しては、前のデータによる影響をうまく表現するために、これもディープラーニングの一種だが、**リカレントニューラルネットワーク**（RNN; Recurrent Neural Network）が提案されている。従来から時系列向けのニューラルネットワークもあったが、最近ではこちらに置き換えられるケースが非常に多くなり、時系列解析のスタンダードとなっている。また、画像と時系列を組み合わせることで、動画としての処理も可能である。

3.3.3 自然言語

そして最後が自然言語である。こちらはテキストを扱うような場合である。冒頭で紹介した通り、自然言語の処理に関しては、単語が数値で言えばいくらなのかといった定量評価が非常に難しく、こちらに関しても特徴量を選択

できるようなニューラルネットワークがとても活躍している。

また、覚えておいていただきたい点として、自然言語データは現場に導入する際の難易度が高い場合がある。それは、文章の生成といった問題設定の時に起こるため、例を交えて紹介する。文章生成の際に、「私は機械学習を勉強しています。」という情報が正解だったとする。このときに、機械学習アルゴリズムで予測した結果が「機械学習を私は勉強しています。」であった際に、人間であれば概ね合っていると判断できるが、機械側では順番が異なっているため、不正解であると判断してしまう。この問題に関して、機械学習側での正解の判断と人間側の正解の判断が異なってしまう。正解率が低いように見えた際にもこのような問題で語順が違うだけであるのか、選ぶ単語も間違っているのかといったところの検証が必要となり、こちらは人間が手動で行う以外、適切に判断できるものがない場合が多い。

第 4 章

ビジネスへの応用

4.1 企画から開発・運用までの全体像

機械学習のプロダクトを開発・運用していくまでの全体像に関して流れを紹介する。機械学習を始めたばかりでは、「モデル構築」と呼ばれる数学とプログラミングを使った箇所についての仕事のみにフォーカスが当たりがちである。つまり、入力と出力（教師）データをクライアントから受け取り、それに対して、その入力と出力をうまく紐づけるようなモデルを作るということである。

しかし、実案件のスコープはモデル構築だけでなく、図 13 に示す通り、遥かにそれよりも広いのである。

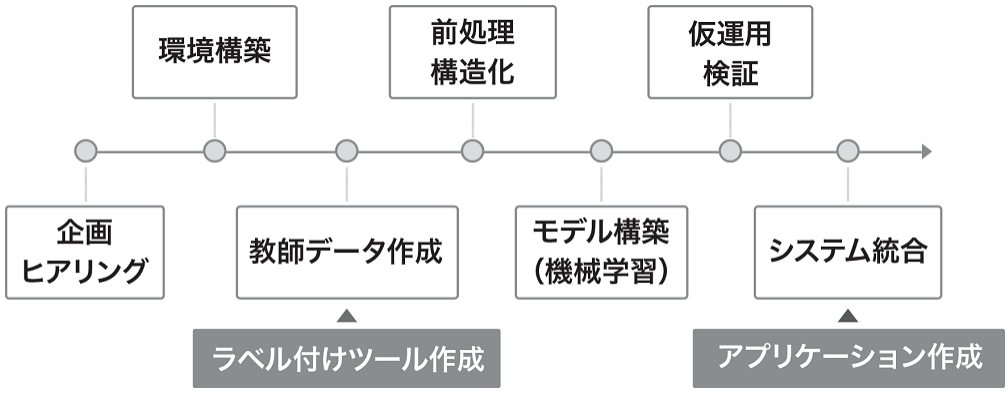


図 13: 企画から開発・運用までの全体像

4.1.1 企画・ヒアリング

発注者側は企画、そして開発者側はその内容のヒアリングを行う。企画・ヒアリングにおいて大事なことは、最終的なシステムの統合を視野に入れながら考えることである。発注者側としては「AI を使いたい」というような曖昧な段階で相談を行うことが多い。その際に相談を受ける側としては、相談内容を実現するためには、どのようなデータが必要であるか、どのようにデータを構造化できるか、どのような手法を選定すると良いか、最終的にシステムに統合する際にはどの程度の精度があれば意思決定できるか、システムの統合向けに保守運用も考慮してどのような技術選定をすべきか、といった事柄を最初の段階で具体的かつ明確に決めておくことが望ましい。途中で変

更になることも多々あるが、最初に視野に入れられなかった事柄を途中から考えることは難しいことに注意がすべきである。

更にこのヒアリングの段階で各工程を細かく、どのような落とし穴があるかも含めて分解し、どの程度の期間がかかるのかといったことを検討する必要がある。大きく時間がかかるポイントとしてはデータ収集や、そのデータ収集の前にある解析環境の構築、教師データ作成である。そのため、どのようなデータが既に入手可能であるのか、そのデータはクライアント側の環境以外でもアクセスできるか、クライアント側でのみアクセスが可能な場合はどのように解析環境構築すればいいか、といったことまで、実際のフローに結びつくように開発側は徹底的にヒアリングする必要がある。また発注側も開発側のヒアリングに備えて、上記の回答を準備しておくと円滑に進められる。一方、発注側視点では、データの受け渡しに関する環境の準備や、そして教師データの準備以外にも、秘密保持契約書等の契約書の準備であったり、上長の許可、また医療分野であれば倫理審査などを通す必要がある。

企画・ヒアリングの際に重要なことは、「技術的な実現可能性」の軸だけで決定せずに、「費用対効果」の軸も考慮した上で開発を行う判断を下すべきである。こちらの判断の方法は AI を使うべき案件の切り分け方で解説する。

4.1.2 環境構築

機械学習の周辺技術の章で解説を行ったが、データへのアクセスまで含めた環境構築を行う。企画段階で選定した技術をベースに環境を構築していくため、この段階で考えることは少ないが、思いの外、環境構築は躓く点が多いため、計画したスケジュール通り順調に進めることが難しい。そのため、開発者側は幅を設けたスケジュールを引く必要があり、発注者側もその点への配慮が必要である。

データベース等へアクセスする機会も増えるため、本番環境のデータベースを誤って変更してしまわないためにも、解析用に複製した安全なデータベースを準備することが望ましい。発注者側はデータが保管されているデータベースで使用されているソフトウェアや VPN (Virtual Private Network) へのアクセス方法などを明確にしたうえで開発者側へ相談をすることが望ましい。

4.1.3 教師データ作成

次に教師データ作成である。 この作業にはデータの整理も含まれている。

演習でも取り組んだが、機械学習を適用していくためには、入力となるデータと教師データがペアで揃っている必要がある。最初の段階でデータは蓄積されているも、教師データはペアになっているケースは多くなく、手動で一からラベル付けを行うこともある。その際には開発者側と相談して効率的に進めることが望ましい。具体的には、枚数が少ない場合はどうにか手動で教師データを作成してしまうと良いが、枚数が多い場合は、別途手間であるが前処理アプリを開発者側で作り、ラベル付けを効率的に進める方が早いケースもある。この教師データ作成は想像以上に労力とコストがかかる作業であり、まず一枚ラベル付けを試しに行ってみて、最終的にどの程度時間がかかるかを算出してから進めると良い。

また、どの程度のサンプル数を確保すれば良いかという疑問も出るが、問題設定によってケースバイケースであるため、一概に算出することができない。そのため、有限な時間内でラベル付けを行い、そのデータをもとに解析を進めてみて、思うような精度が得られなければ、また教師データ作成に戻ってくることになる。スケジュールを立

てることが難しいため、3 カ月のように期間を決めて PoC (Proof of Concept) を実施して、その結果によってプロジェクトの続行もしくは中止を選択することがよく行われる。

4.1.4 前処理・構造化

環境構築や教師データ作成が完了し、いざ機械学習によるモデル構築に取り掛かろうとする際、主に開発者側であるが、文章や画像などをどのように機械学習の入力としてベクトル化して良いかわからない問題がある。このように文章や画像といった非構造データを構造化することに関する知識が必要となる。

そして、この構造化と同じタイミングで、外れ値除去やスケーリングなど、問題設定に応じたデータの前処理が必要となる。画像であれば何でもディープラーニングを使えば良いというものではない。とりあえずディープラーニングを使用するも思うような精度が出ず、その原因を探りながら、結果的にデータの前処理を行っていくのである。 AI に任せればすべてがデータに基づいて解決されると思っていたという楽観的な意見を聞くこともあるがそうではない。現場での解析は、モデルを作りながら、どのような原理になっているかを人間が考察を繰り返すことで理解し、そのノウハウを反映していくことで精度が向上していくのである。

4.1.5 モデル構築（機械学習）

モデル構築は開発者側の作業となるため、発注者側ではディープラーニングを使用してほしいなど特定の要件は事前に依頼しておくべきである。また、機械学習のアルゴリズムの章で紹介したように、各アルゴリズムで長所と短所があるため、運用の際にその短所が問題とならないようなアルゴリズムを選択すべきである。特に注意すべきは推論を行う計算機での計算時間である。学習用のサーバーでは高性能な計算機環境を準備できるも、実運用の際の推論に使用する計算機はスペックが低く、想定よりも推論のスピードが遅くて使うことができず、手法の選定からやり直しとなることも想像できる。そのため、計算機環境は学習の場合だけでなく、運用で使用する推論の場合にも目を向けておくことが大事である。

このモデル構築に入る前に、依頼する際の納品物に関して発注側と開発側の認識のすり合わせを行っておく必要がある。モデル構築の最終的なゴールは、与えられたデータを訓練と検証データに分け、その検証データに対する予測精度がどの程度になるかを検証することである。そのためこの段階での納品物はレポートとなることが多い。このレポートの段階で、発注者側が納得すれば、次の仮運用やシステム統合に移行する。

発注先の選定基準としては、すでに類似案件を経験しているかが重要である。すでに経験があれば、前処理等のノウハウがたまっていたり、別の類似案件の学習済みモデルを使用できる場合がある。類似案件の学習済みモデルをベースにチューニングすることで、準備できるデータの量がたとえ少量でも、ある程度良い精度が出せる場合が多くある。これを**転移学習**といい、ディープラーニングでは**ファインチューニング**という技術として知られている。そのため、経験面や実装面ともに、類似案件の経験があるかは重要な選択基準となるのである。

4.1.6 仮運用・検証

モデル構築が上手くいった場合でも、過去のデータに基づいて学習した結果であり、実際に運用をはじめると、検証結果より悪い結果に遭遇することが多々ある。そのためモデル構築後も、どの程度の結果が出るのかを仮運用しながら検証していく必要がある。

特に事前に知識のない発注者の場合、検証というとこのモデル構築の際の検証データに関する検証をさすことで納得できる。しかし、データ解析側でどの部分を検証データにするのかといった選択や何パーセントに設定するのかといった選択を恣意的におこなうことができるため、良い結果が出るような検証データの選び方に工夫している可能性もある。そのため、検証結果が良かったとして、運用フェーズでうまくいくとは限らず、実際に仮運用しながら検証してみることが重要である。

ここで仮運用しながら検証を行うと、過去のデータに対する予測値を計算していた仕組みとは異なり、定期的に予測値を計算すると言った仕組みや、誰でも使えるようにアプリケーションに落とし込んでいく工程が発生する。この点に関しては、データ解析とはまた異なるアプリケーションを作っていくような技術が必要となり、この段階で開発コストも上乗せとなる。そのため仮運用を始めてからでは費用対効果が合わないケースもあるため、モデル構築の段階でレポート納品としてまずマイルストーン置き、仮運用に進めるかを判断できるようにする。仮運用を行いながら、もしうまくいかないケースであればモデル構築や前処理から再度見直していくことになる。

4.1.7 システム統合

仮運用がうまくいった際には、開発したものを既存のシステムに統合する工程がある。既存のシステムがない場合は新しく Web アプリケーションを作ることもある。

この際に、発注者側としては既存のシステムがどういう仕組みで作られているのか、どの業者によって作られているのかを考えておく必要がある。特に医療系ではベンダーが限られており、自前で作ったシステムを他の医師が使えるように統合するということは容易ではない。また PMDA と言った審査を通すことが必要になることもあり、最終的にシステムに統合できるのかを把握した上でプロジェクトを開始しなければならない。

4.2 AI を使うべき案件の切り分け方

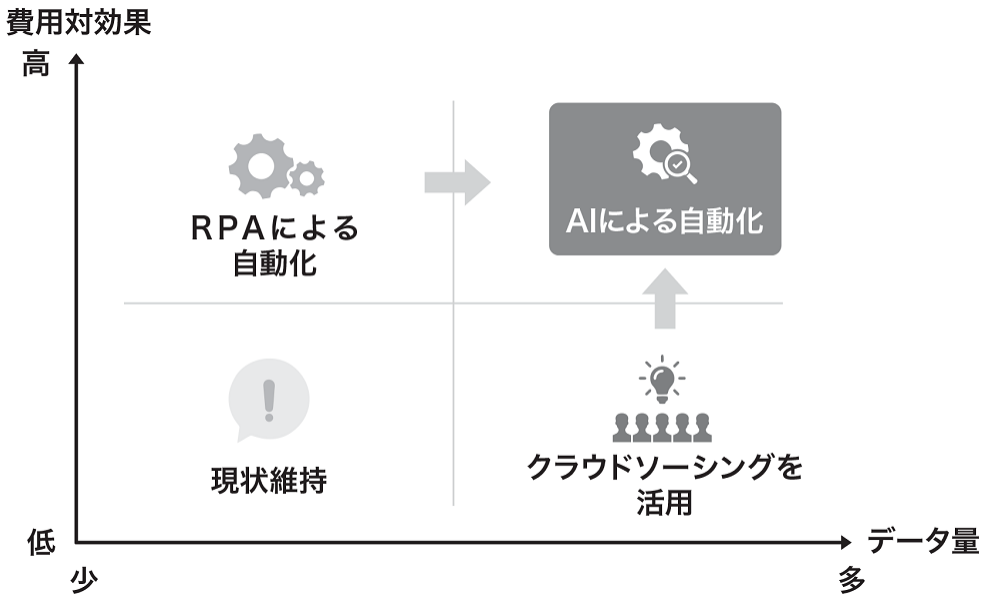
費用対効果とデータの整理状況から、AI を使うべき案件の切り分け方について 4 象限に分けて紹介する。

図 14: 使うべき技術の判断基準

この 4 象限に分ける際には、2 つの軸が存在する。1 つ目はデータ量、そしてもう 1 つは費用対効果である。

4.2.1 データ量が少なく、費用対効果も低いケース

この場合、無理に開発をしたところで、開発的な問題、そして発注側も完成したシステムによって受ける恩恵の少なさから、作らないほうが良かったという結論になる可能性が高い。そのため、この場合は、現状維持が最も良い



選択である。

4.2.2 データ量は多いが、費用対効果が低いケース

事務作業が主な事例としてよく挙げられる。データ量が多くかつ単純作業として、事務作業を自動化したいといった相談も多くあり、蓄積されているデータ量からすると実現可能性は高いが、もう一つの軸である費用対効果も考慮した上で判断すべきである。

たとえば、Fax で送られてきた書類から文字を抽出して自動的に請求書のフォーマットに書き起こすシステムを開発することとする。この開発したシステムの完成によって、事務作業員 1 人分の 30 万円が毎月支払わなく良くなるが、その代わり、開発費用 1000 万円、保守運用で毎月 10 万円必要となるとすれば、開発費用を回収することが困難もしくは長い時間がかかる。また、AI による精度が 100% は出ない問題から、事務作業を完全に自動化することは難しく、事務作業員のフォローが必要となり、そこまで劇的な人件費削減には繋がらない。

そこで、このようなケースに関しては、まずクラウドソーシングを活用したコスト削減が効果的である。その際に工夫しておくと良い点として、クラウドソーシングで得られた結果を将来的に機械学習に使用できるように入力と教師データを 1 セットで取得できるようにしておく。この工夫によって、データがたくさん溜まった際に、AI による自動化の検討を行える。そして、クラウドソーシングに投げている仕事の一部を自動化してコスト削減につなげることができる。

4.2.3 データ量は少ないが、費用対効果は高いケース

データ量は少ないが費用対効果が高く、そしてロジックがある程度人間のノウハウとして明確になっているケースがある。このような状況では、**RPA（Robotic Process Automation）**による自動化の検討が良い。 RPA は単純に人間の行っていた作業をプログラミングし、再現可能にしていく技術である。RPA に関する開発を行う企業も多

く出現しており、ロジックさえある程度確定していれば、費用対効果も高いため開発コストに見合うような結果を得ることができる。

この RPA を実施していく中で、ロジックを言語化することが難しいことがある。その際には機械学習を用いてそのロジックをデータ駆動で見つけ出していくといたことが望ましい。RPA を使用しながら、適宜必要な箇所は機械学習に落とし込むことにより、費用対効果も高く、データの整理状況も徐々に高くなっていくようなシステムを作っていくことができる。

データ量が多く、費用対効果も高いケース

実は最初から当てはまるケースは少ない。ステップとしてクラウドソーシングもしくは RPA による自動化を検討し、データ量と費用対効果の軸の両方をクリアできた際に、次のステップでは AI によって自動化を検討すると良い。マイルストーン置くことで、開発側は焦らずに開発でき、そして発注者側としても、効果を短期間で得ることができるといったような関係性を築き上げることができる。

4.3 外注と内製化

案件を進めていく上で、開発を外注して行うのか、それとも自社内で開発して内製化していくのか最初の選択がある。外注と内製化のそれぞれに関するメリット、デメリットを考えていく。

4.3.1 外注

まず外注のメリットとしては、データ解析を行う人を確保する必要がないことである。AI 人材は不足していると言われており、採用が難しいといった問題がある。そのため、開発しようにもまず人手の確保が優先であり、開発しようにも採用できずに始められないといった問題が生じる。そのため、このような人材の確保が必要とならない場合、案件の企画さえ固めてしまえば、外注先を見つけることができれば、即時に開始することが出来る。

一方、デメリットとしては、開発されている機械学習やシステムの仕組みに関して把握できていないため、修正が必要な場合にも依頼を行わなければいけないというメンテナンス性の問題がある。また、開発後の契約期間が切れてしまうと修正することが困難となり、**技術的負債**が残る。この技術的負債は発注者側にとって大きなデメリットになる。

4.3.2 内製化

内製化を行うメリット・デメリットは、外注のメリット・デメリットの反対である。開発を行う際の修正が比較的容易であり、その分の開発できる人材が必要となるため、案件を開始する前にまず人材の採用が不可欠となる。また、人材を採用した後も育成ができる環境が必要となる。エンジニアは技術に対する向上心が高いため、技術的なチャレンジが少ないと転職する傾向にあり、そのためにも育成を行っていくシステムが必要である。

最初から採用や育成ができる環境を作り上げることは容易ではない。そのため、データ解析の案件を行う際は、ま

ず最初に費用対効果がでるのかといったところの検証を外注して行う。そして必要であれば、その案件を引き継げるような人材を時間をかけて確保しながら育成していく。最終的にはその人材に案件の詳細を把握してもらうというと流れが望ましいと言える。