Machine Learning in Business

Table of Contents

Introduction

本書は、「機械学習をビジネスに活かすための実用書」を目指して書かれています。
具体的には、以下のような点について重点的に記載を行っています。

- 導入検討：機械学習をビジネスへ適用する際の検討プロセス
- 設計開発：既存システムに機械学習を組み込むための、設計及び実装方法
- 運用監視：機械学習を継続的に運用していくための運用・監視方針

逆に、以下の点については記載をしていません。

- 理論解説：機械学習の、理論的な説明
- 実装解説：機械学習アルゴリズムを実装するための、詳細なチュートリアル

これらの点については、良書・また偉大な Coursera の Machine Learning Course があるため、そちらを参照いただければと思います。そもそも本書を書くきっかけとなったのは、こうした「機械学習の理論・および実装」と、「ビジネスへの適用」に大きな隔たりがあると感じたためです。
機械学習をはじめとしたいわゆる「人工知能」に対する期待は過度に煽られていると感じますし、また機械学習の実装として示されるようなコードはサンプル、あるいは研究用に過ぎないことが多く、実際に本番システムの中に組み込むには難しいものになっています。

本書は、こうしたギャップを克服し「適切な導入検討プロセスの下に、実運用可能な形で機械学習を組み込んだシステムを開発できるようになる」ための手引きを目指しています。

解説の流れは、以下のようになっています。

- Chapter1：企画者・開発者双方が押さえておくべき機械学習の基礎知識
- Chapter2：実際に機械学習の導入を検討する際の、検討プロセス
- Chapter3：開発者が押さえておくべき、 機械学習の設計・実装・テストの考え方

Chapter1/Chapter2 は機械学習の導入を検討するに当たり、企画者、開発者双方で共有しておくべき基礎知識と、プロジェクトの推進方法について記載しています。これらは、プロジェクト時にいつでも参照できるハンドブックとして機能するようにしています。
Chapter3 は完全に開発者向けで、機械学習を組み込んだシステムを開発する際の設計、実装、またテストについて記載を行っています。

全てを合わせれば、「適切な導入検討プロセス」と「実運用可能な形での実装方法」が理解できるのではないかと思います。

本書を契機として、一件でも多く有効な機械学習の導入がなされればと願っています。

Chapter1

機械学習とは何かを知る

本章では、機械学習の導入を検討するにあたり押さえておくべき、基本的な知識について解説していきます。

本章を読むことで、以下の点について理解することができます。

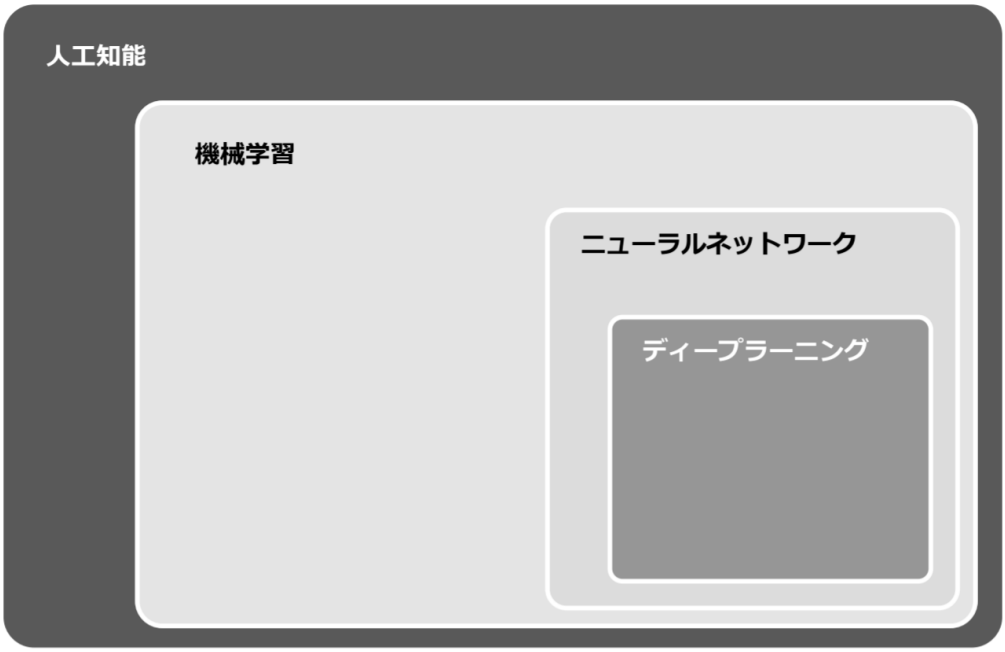
- 機械学習と人工知能、ディープラーニングの関係
- 機械学習はどのような仕組みで動いているのか
- 機械学習の適用領域
- 機械学習の実利用例
- 機械学習を実装する際の選択肢

これらを企画側・開発側双方で共有し、機械学習に対し互いに過度に楽観的・悲観的にならずに検討を進めていけるようになることが本章の狙いです。

機械学習にまつわる、用語の整理

機械学習は「人工知能」やディープラーニングといったキーワードとないまぜで使われることが多いため、まずこれらの関係を整理しておきます。

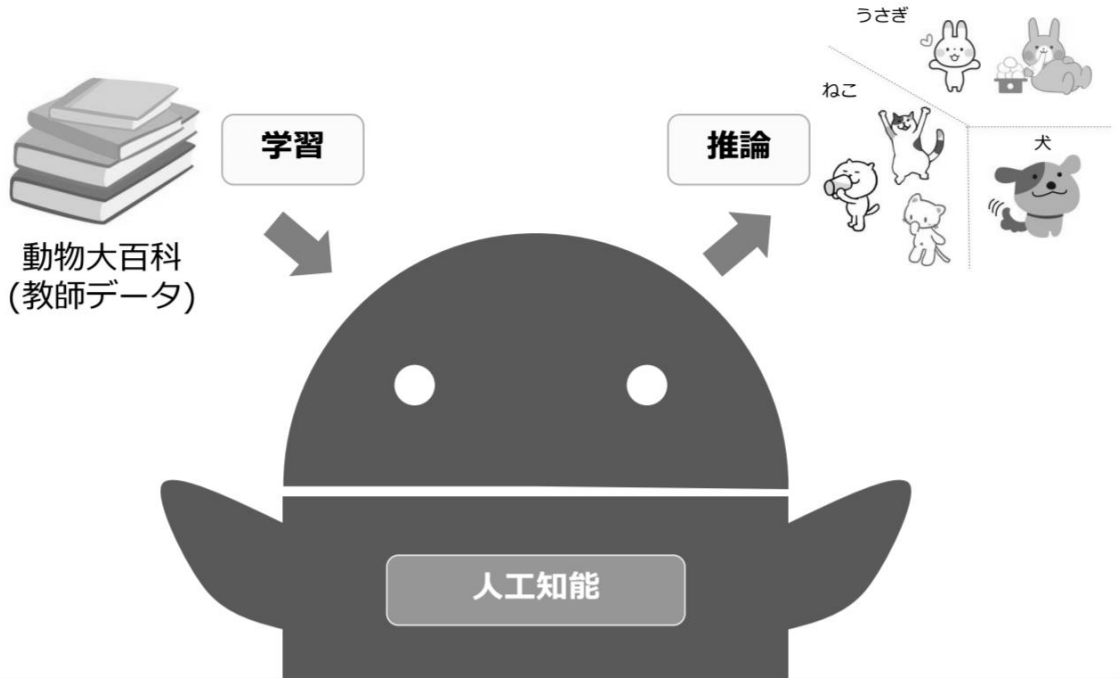
機械学習周辺のキーワードの関係を図示したものが、以下になります。



- 人工知能が、最も大きい枠組みになります
- 機械学習は、人工知能を実現する手段のうちの一つです。
- ディープラーニングは、機械学習の手法のうちの一つです。

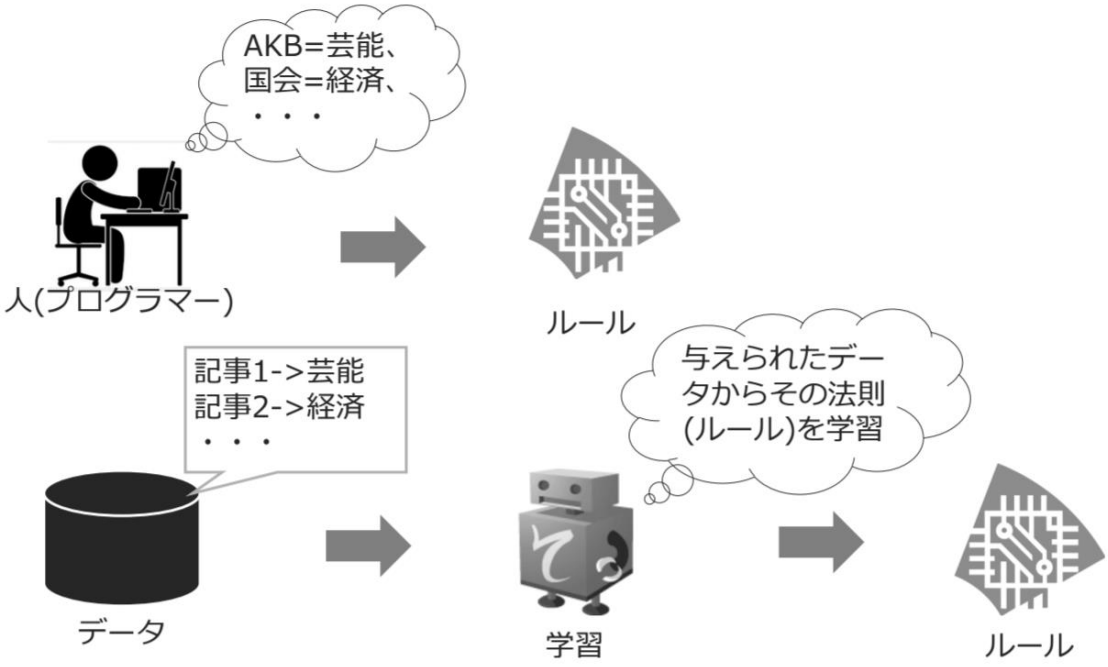
人工知能

人工知能とは、「学習」した結果を基に「推論」を行うことができるシステムです。



通常のシステムは、この「推論」を行うために必要なルールやアルゴリズムを人があらかじめシステムにプログラムします。これに対し、人工知能はデータのみを与えることで、データに即したルールやアルゴリズムを「学習」します。この違いを示したものが、下図になります。

例：ニュース記事を分類する例

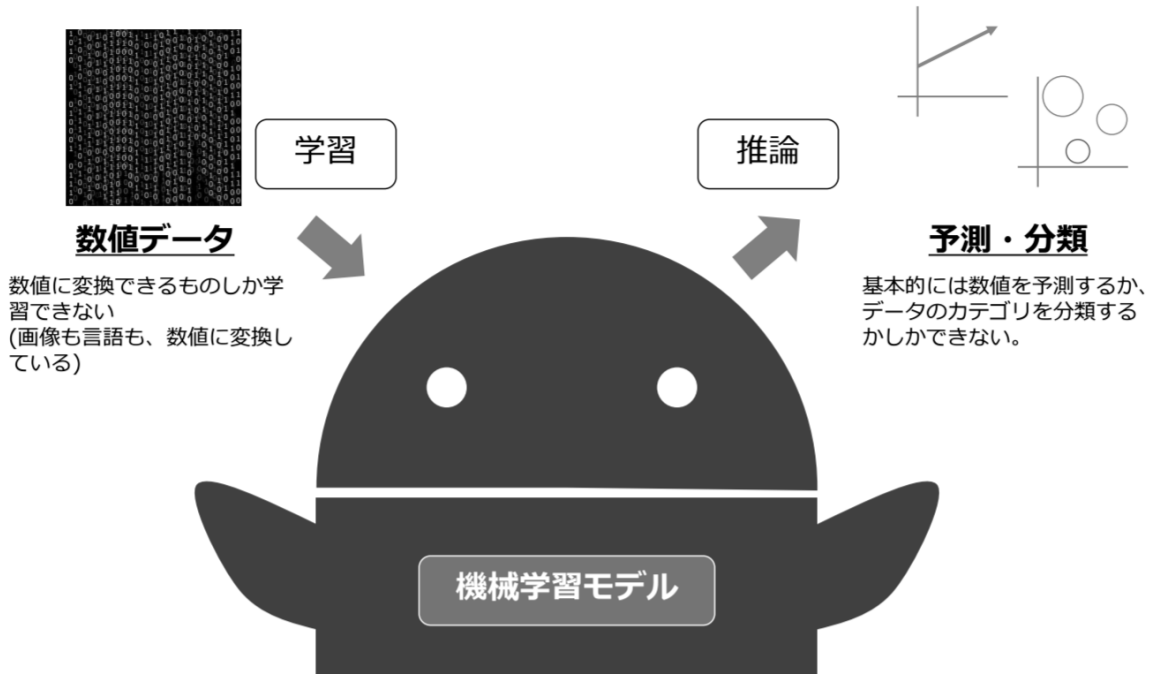


- 通常は、人が分類のルールをプログラミングする
- 人工知能は、与えられたデータからその中に潜む関係性を学習する。それで分類を行う

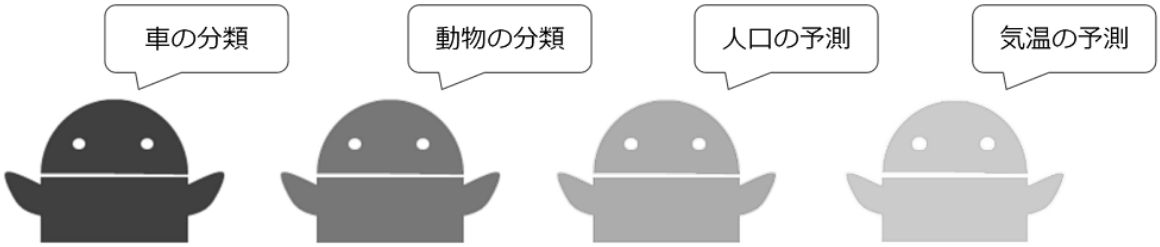
この人工知能を実現するための技術の一つが、「機械学習」になります。

機械学習

人工知能は「学習」し「推論」するシステムでした。機械学習も基本は同じです。
しかし、機械学習で学習できるのは「数値データ」のみであり、推論できるのは基本的に「値の予測」か「分類」になります。



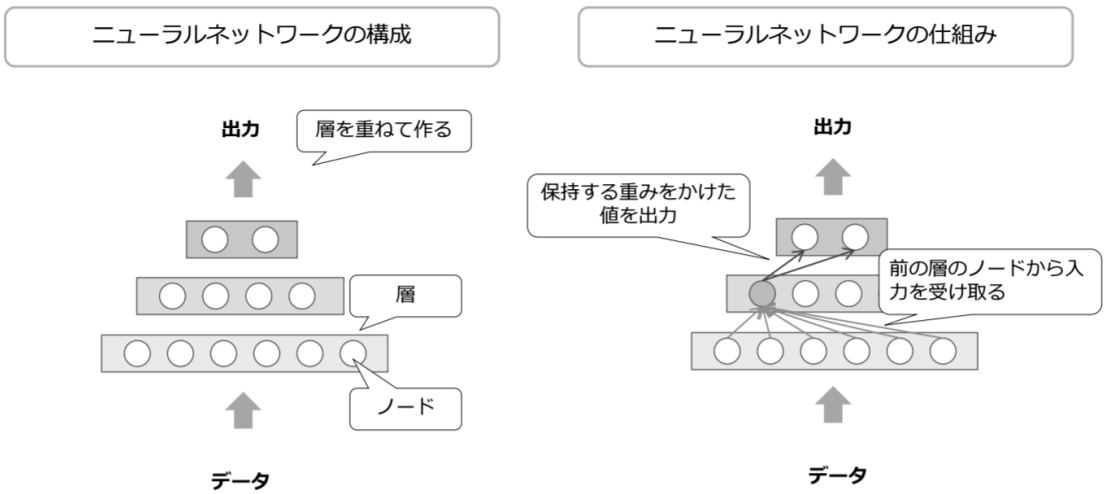
画像や言語といったものも、機械学習を適用する際は何らかの形で数値化を行っています。
また、一つの機械学習モデルは基本的に一つの推論しか扱えません。



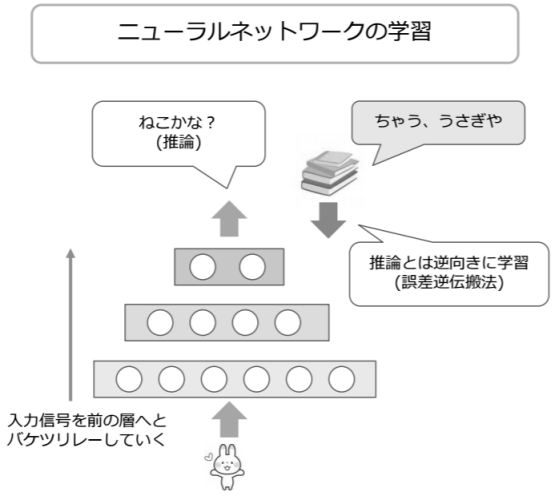
よって、**単一の、なんでもできる機械学習モデル(人工知能)が存在する**というのは誤りです。 何を推論させたいかによって、どんなモデルを使うのか、どのようにデータを数値に変換するかなどは変わってきます。「ディープラーニング」は、この機械学習のモデルのうちの一つであるニューラルネットワークの特殊なパターンになります。

ディープラーニング

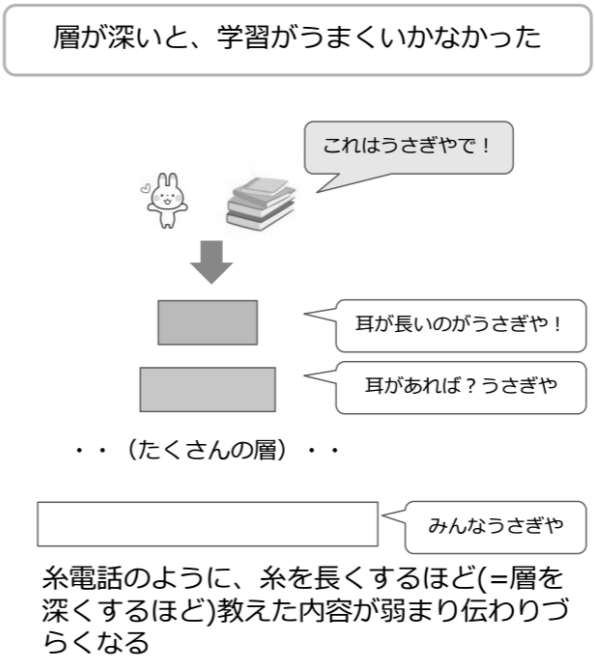
ニューラルネットワークとは、ノードを集めた層(レイヤ)を積み重ねて作るモデルです。



各ノードは、前の層のノードから入力を受け、次の層のノードへ出力を行います。これを繰り返していくことで最終的な出力へとたどり着きます。 最終的な出力が望ましい値になるように、各層内のノードの重みを調整することが、「ニューラルネットワークの学習」となります。



そして、ニューラルネットワークは層を積み重ね、ノードを増やすほど表現力が増します(表現できる組み合わせのパターンが増えるためです)。 しかし、層を増やすほど学習が上手くいかないという課題が長らくありました。



これを解決したのがディープラーニングです。解決した、とはいえ新しい手法が組み込まれたというよりは、膨大なデータセットを扱えるコンピューティングリソースが手に入るようになってきたというのが理由としては大きいです。つまり、層を深くして教えた内容が薄まるとしても、それを補うほどの量のデータを学習させる、ということです。その意味で、一般的にディープラーニングは他の機械学習モデルよりも学習に多くのデータ、またコンピューティングリソースを

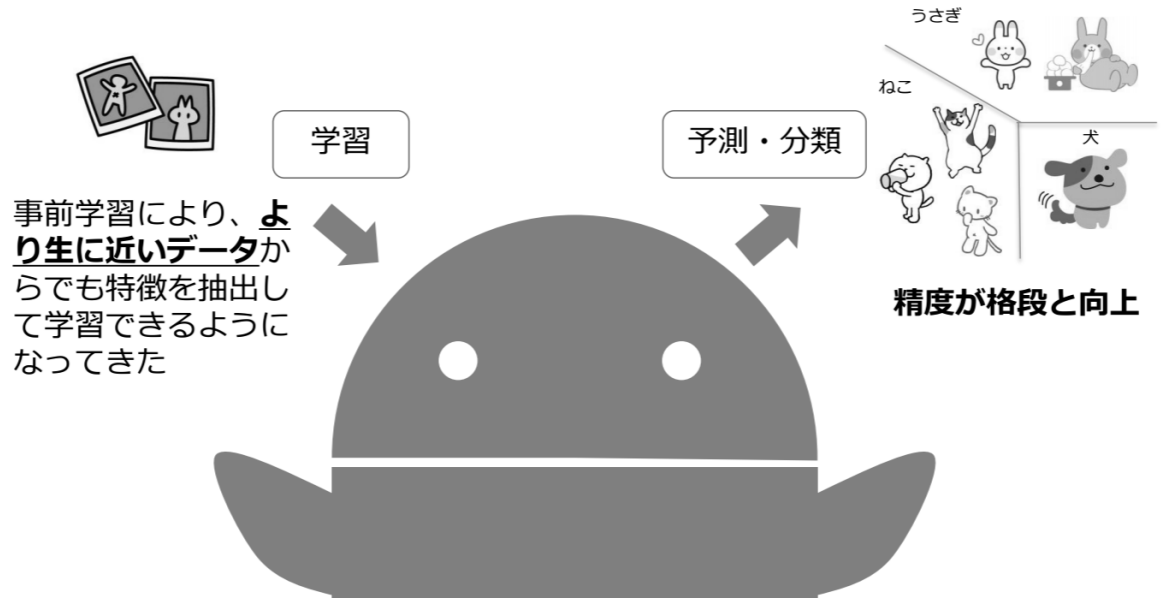
必要とします。その分多層にしたネットワークにより複雑な法則を獲得することができ、人間が明確に定義できない「なんとなく」感じているような特徴もある程度学習できます(これが(人が明示しなくても)特徴を自動的に学習する、と言われるゆえんです)。

ただ、ディープラーニングにおいても学習させるのは人間の役割である点に注意してください。

どんなに優秀な生徒(ディープラーニング)でも、先生(人間)がいなくては学ぶことができません。さらにいえば、先生(人間)側が優秀でなければ生徒(ディープラーニング)の力を発揮させることもできません。
そのため、ディープラーニングが登場したから人間の役割が不要になるということはなく、むしろどう学習させるかはまだ課題が多いということは念頭に置いていただければと思います。

まとめると、ディープラーニングの登場により変化が起こったのは以下点に集約できます。

- あえて教えなくても人が感じている特徴を学習できるようになったため、より(人間が触れている)生に近いデータも学習データとして扱えるようになった
- 多層に重ねたネットワーク内の膨大なパラメーターにより、精度が格段に向上した



ただ、「学習をさせる」のはまだ人間であり、また何より基本的には単一の推論しか行えない点は変わっていません。

逆に言えば、ディープラーニングの登場により勝手に学習を行うようになる、人間が想定しなかったいろんなことを推論し始めて果ては人間を・・・みたいなことは起こりえません。

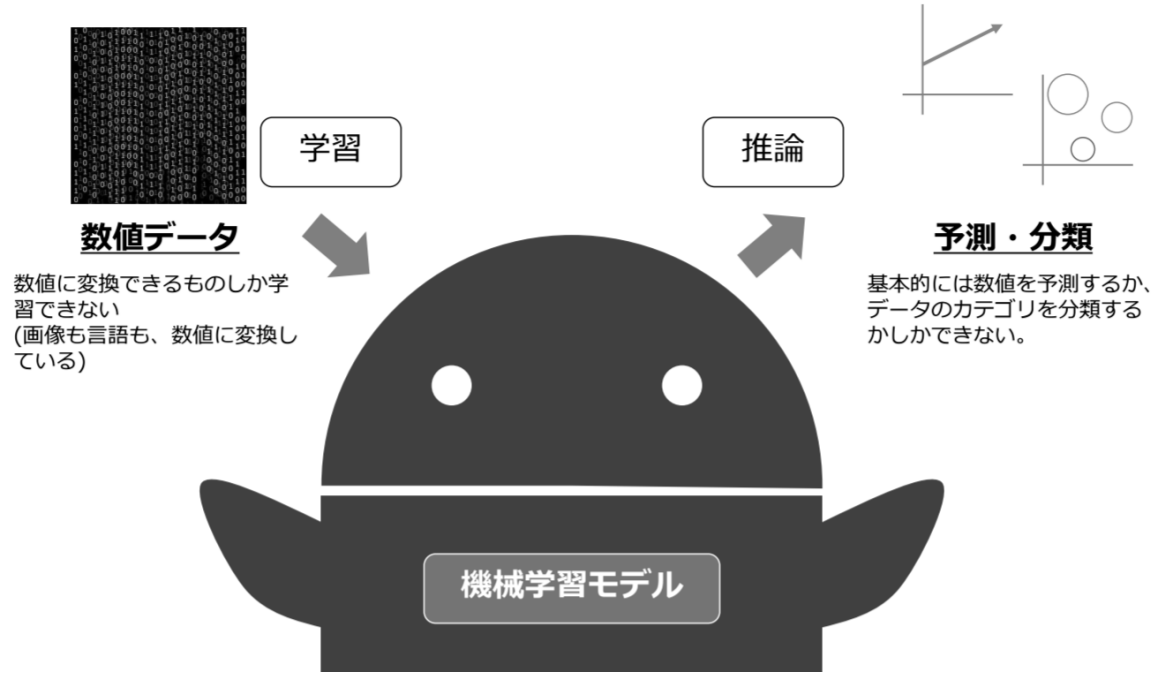
何を学習させ、何を推論させるかは人間が定義する

この点には変わりがないということを、押さえておいていただければと思います。

機械学習の仕組み

機械学習が実際どのように「学習」し、「推論」するのか、ここではもう少し中の仕組みを見ていきたいと思います。中の仕組みというと数式が出てくるのではないかと警戒する方もいるかもしれませんが、数式は出てこないので安心してください。

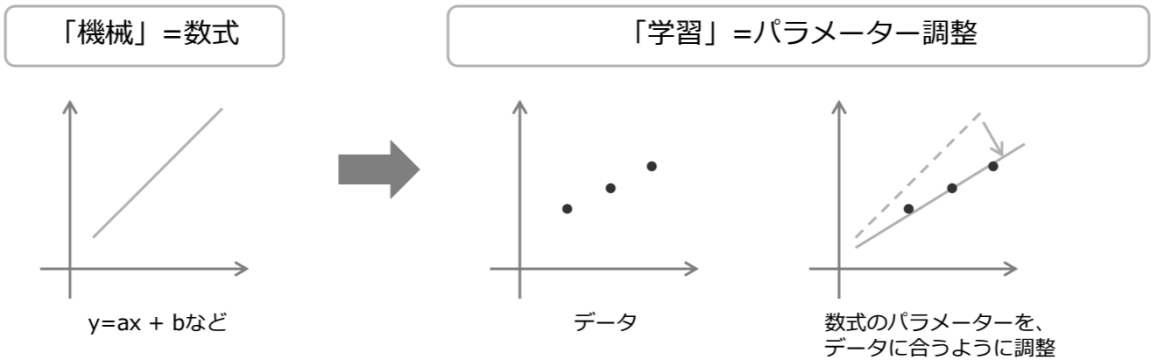
まずは、機械学習の仕組みのおさらいをします。



この図から、機械学習モデルが行っているのは 「数値データ」から「予測・分類」を行う ことであるというのがわかります。数値を処理して結果を出力するわけですから、つまり機械学習モデルの実体はまさに「数式」であることがわかります。

数式は y = ax + b のようなものですが、この式で傾きを表す a や切片を表す b があるように、数式には様々なパラメーターが存在します。 この「パラメーターを、データに合うよう調整」することがまさに学習に該当します。

イメージ的には、以下のようになります。



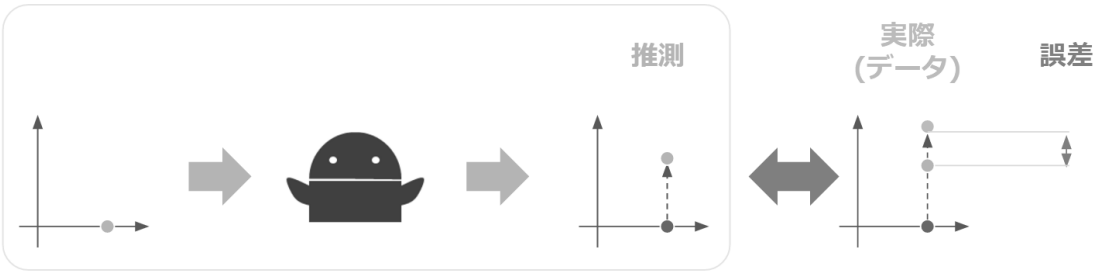
機械(=数式)を、データに合わせて学習(=パラメーター調整)させる。

これが機械学習の意味するところになります。

学習のさせ方

学習は、データに合うようにパラメーターを調整することでした。

この「データに合う」という状態を具体的に表すと、データの中の事実とモデルからの推測の間の「誤差」が小さい状態と表現できます。



よって、機械学習における「データ」とは、「x ならy」という関係性を含んだデータであることが前提になります。　逆に言えば、関係性が含まれないデータは機械学習で扱うことはできません。

例えば、家賃、というデータに関係がある項目としては、間取りや駅からの近さが思い浮かびます。逆に、その物件の名前や担当の販売員とは、何の関係もなさそうです。

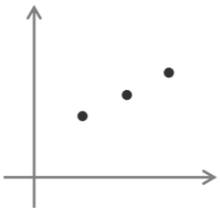
- 関係性が含まれる: 2LD、駅から徒歩 2 分→家賃 80,000/月
- 関係性が含まれない: 物件名メゾン一号、販売員 A→家賃 80,000/月？？

このように「関係性が含まれない」データの場合、機械学習で扱うことはできません(無理やり入れることは不可能ではありませんが、精度が出ません)。

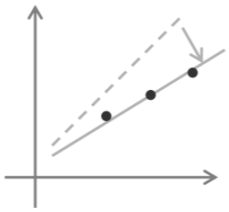
この関係性は、明示的に教える方法と関係性そのものを推測させる方法の 2 種類があります。 それが「教師あり学習」と「教師なし学習」です。

教師有り学習

実際のデータ=教師



データと予測の誤差を調整

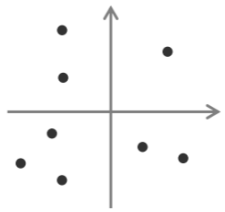


AならB、というデータを与え、それに合うようにモデル(=数式)のパラメーターを調整する

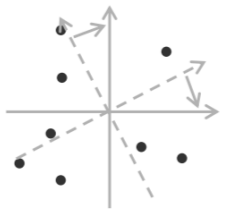
「合う」というのは、データとモデルの予測値の間の「誤差」を小さくするということ。
つまり、パラメーター調整=データと予測の間の誤差の最小化。

教師無し学習

実際のデータ



データと予測の誤差を調整

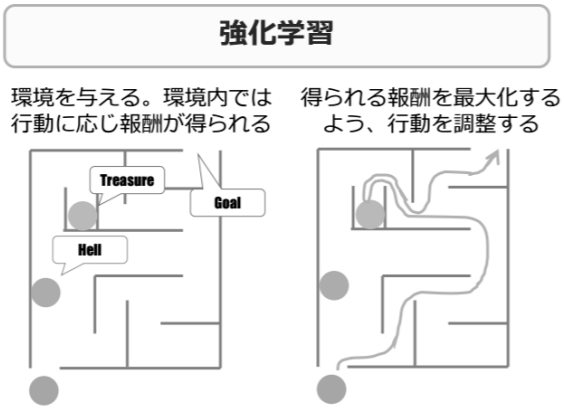


データ間の関係性をうまく説明できるよう、パラメーターを調整する。

「うまく説明できる」というのは、モデルが示す関係性(AとBは同じカテゴリ、など)と実際のデータ上の関係性の間の「誤差」を小さくすること。
この誤差の最小化が、「パラメーターの調整」となる。

教師なし学習では関係性そのものを推測させることが可能ですが、もともとのデータに関係性がなければうまくいかないのは上記の通りです。 また、関係性そのものを推測させる関係上、人間が推測する関係性とはずれてしまうこともあります。 そのため、関係性を明示する必要がないからといって安易に教師なし学習を行うのはあまり実のある結果にはつながりません。

これ以外に、「行動」とそれによって得られる「報酬」の関係性を学習させる手法もあります。 それが強化学習と呼ばれる手法です。



プロ棋士を指し負かしたと話題になった AlphaGo は、この強化学習を利用してある状況における「指し手」と「報酬」の関係性を学習しています。

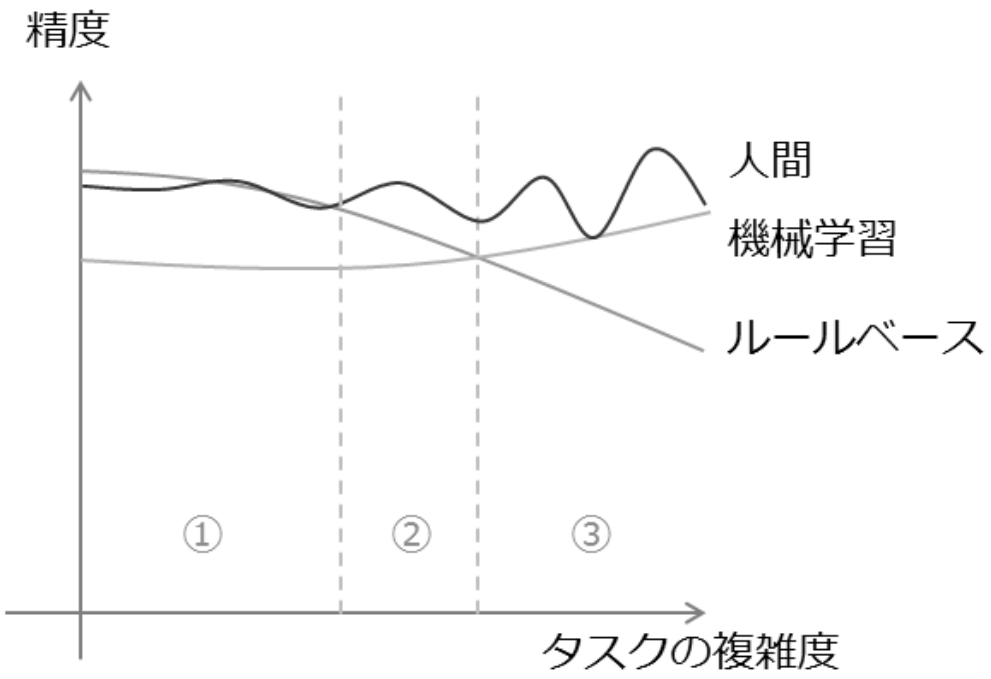
いずれにせよ、機械学習はデータの中にある関係性を推測するための技術であり、与えるデータには(推測可能な)関係性がなければならない、という点を押さえておいてください。
端的には、人間にも推測不可能であれば機械学習でうまくいくということはありません。囲碁の手の学習も、いい手と悪い手を判断できるからこそ、学習が可能なわけです。

そのため、機械学習を利用する際はまずデータを見て、その関係性が人なら判断できるか。そのチェックが学習の第一歩となります。この点については、後の章の機械学習の導入プロセスにて詳しく触れたいと思います。

機械学習の適用領域

機械学習は、既存のシステムの上位互換である万能なシステムというわけではありません。 ここでは、機械学習はどのような領域で力を発揮しうるのかについてみていきます。

下図は、人間・機械学習・ルールベース(いわゆる既存のプログラム)それぞれについて、タスクの複雑度と精度を簡単に表したものです。



- 人間は、どんなタスクでも割合高い精度でこなすことができます。しかし、当日の体調や気分などに左右されることも多いです。
- ルールベース(既存のシステム)は、ルールが定義できることが可能な領域では、人より正確に動作します(これは機械の特性そのままです)。しかし、ルールで定義しきれない、感覚的・複雑なタスクになるほど対応できなくなります。
- 機械学習は、ルールが定義可能な領域ではそれをきちんと実装したシステムに勝つことはありません。しかし、複雑なタスクにおいてはルールベースよりも柔軟に対応できます

上図のなかでは、1,2,3 という 3 つの領域を設けています。この領域には、それぞれ以下のような意味合いがあります。

1. タスクは決められたルールに沿って行われており、ルールをプログラム化する既存のシステムで十分対応できる
2. ルールで定義することにだんだんと無理が出てきて、人がシステムから漏れてきたタスクを補完するようになる
3. 機械学習が、ルールを十分に含んだシステムより優位な性能を出すことができる領域

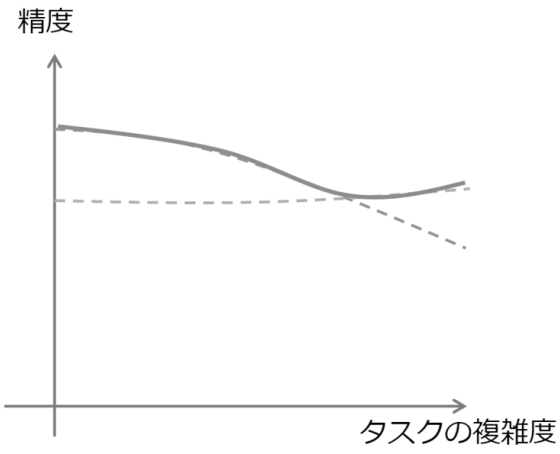
多くの企業では、システムだけで業務が完結しているということはありません。 システムからのデータを Excel で分析して判断したり、システムで設定された項目を人が直したり・・・ということが往々にして行われています。 こうした作業が顕著になるのが「2」のフェーズになります。

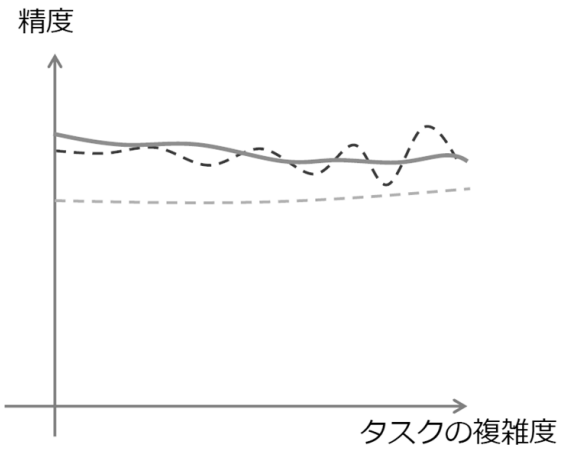
一方、「2」のフェーズでは機械学習のシステムよりも既存システムの方が精度が高い点に注意してください。 十分なルールが定義されたシステムは、一般的に機械学習よりも高い精度を発揮します。 よって、機械学習の導入に際しては「既存のシステムより優位な結果を出しうるか」、つまり「3」のフェーズにあるかどうかの見極めが重要となります。

逆に言えば、「3」の領域以外では、機械学習の出番はありません。 「2」になるか「3」になるかはデータの質や量によって大きく変動します。 よって、機械学習の導入に際しては「3」に到達しているかの確認が重要になります。この点については、後段の導入プロセスで詳細に解説していきます。

この「3」の領域で機械学習が果たせる役割は、以下の 2 点になります。

- 既存のシステムと協調させることで、複雑なタスクにも対応できるようにする
- 体調などで変動しがちな人の精度をサポートする





先ほどの図からもわかる通り、機械学習が活躍できる「3」の領域自体はそれほど大きいものではありません。 仕事全体では、既存のシステムや人間で行うほうが優位な作業のほうが圧倒的に多いわけです。 そのため、これらの「置き換え」ではなく、「協調させて」導入するほうが、全体として高い効果を出すことができます (仮に「置き換え」てしまった場合、「1」や「2」の領域で不都合が出てくることになります)。

実際よく機械学習が利用されるレコメンドシステムなども、システムの大半は商品の在庫管理・発注・発送管理など、多くの「ルールを基にした既存システム」が担っており、機械学習が適用されるレコメンドはそのうちの「ルール化が難しい」一部となります。
医療の画像診断などにも最近は応用が進んでいますが、これはまさに人(医師)の判断のサポートとして働いています。 しかし、そうして働くには精度が十分、つまり「3」のステージである必要があります。

- 該当のタスクは、機械学習が活躍可能なステージであるか
- 「置き換え」ではなく、「補完」「協調」的な役割を担わせる

上記の点が、機械学習が活用できる領域の条件であるといえます。

機械学習の実利用例

これまでは機械学習それ自体の解説を行ってきましたが、ここでは実際の適用例を見ていきたいと思います。 適用例については、ビジネスシーンだけでなく、アートシーンにおけるものも紹介したいと思います。

ビジネスシーン

ビジネスにおける貢献とは、端的には売上の向上かコスト削減への寄与の 2 種類があります。
売上の向上の場合、さらに既存サービスへの付加価値か、新規ビジネスかの 2 つに分けられます。

- 売上の向上
  - 付加価値の向上
  - 新規ビジネスの創出
- コスト削減

この観点で適用例をまとめたものが、下図になります。

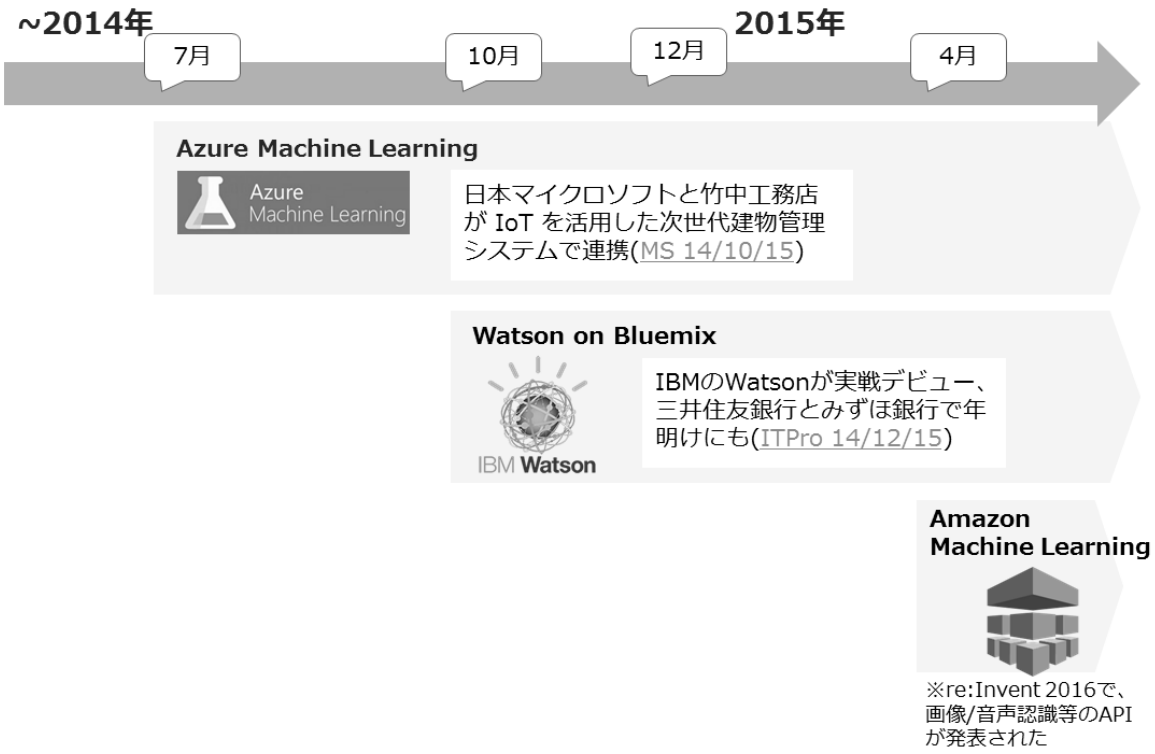
Amazon のレコメンドは非常に代表的な例ですが、Amazon がレコメンドをはじめたのはだいぶ前からです(特許の出願は 1998 年)。 これ以外にも、データ活用の先駆けともいえるコマツの

KOMTRAX は 2001 年から、 またお掃除ロボットのルンバは 2002 年が初号機と、著名な事例の出自は意外と過去からのものです。

このことからもわかる通り、「機械学習」自体はそれほど最新の技術ではなく、ビジネスへの応用も既に行われています。 逆に言えば「ディープラーニングが登場したからビジネスに適用できるようになった」というわけではありませんし、 ディープラーニングを使わなければビジネスへの適用は出来ないわけではありません。 活用はそれ以前から行われてきたわけですし、実際一般的なモデルでも十分有用です。ただ、最近になって活用事例が増えていることも事実です。この要因について、少し見ておきたいと思います。

活用事例が増えている背景には、機械学習そのものの技術的な進歩というより、「機械学習のプラットフォーム化」が大きく影響しています。

端的に言えば、手軽に使えるサービスが登場してきたため、ということです。



2014 年 7 月の Azure Machine Learning を皮切りに、IBM の Watson、Amazon、また Google など大手クラウドベンダが続々と機械学習が扱えるサービスをリリースしました。

ただ、今までにもデータ分析、また予測を行う統計解析のソフトなどはありました。これらと続々とリリースされたサービスが異なる点は、それらが「クラウド的」であるという点につきます。クラウド的というのは、以下のような点です。

- すぐに使い始められる
- 使う機能は、用意されているものから選択する(バイキング形式)
- 費用は、使った分だけ

この特徴により、特定の機械学習の機能を手軽に試せるようになりました。 そのため、既存のパッケージソリューション(BI ツールや ERP など)に比べ、これらは図中にあるように、事例が出るまでのスピードが早くなっています。

また、機械学習の実装に使用されるライブラリも多く公開されてきています。

多くがオープンソースとして使用できるほか、Google など企業から公開されたものもあります。こうしたオープンソースのライブラリの登場により、研究者とサービス開発者双方で共通のライブラリを利用しているという状況が発生しやすくなりました。 つまり、最新の研究が実ビジネスに応用されるまでの時間が非常に早くなっているということです。
Prisma というアプリは撮影した写真に絵画のようなエフェクトをかけることができるアプリですが、 これは Google が発表した Neural Style Transfer という技術を用いています。そして、論文発表からアプリ公開まで、1 年あるかないかくらいでリリースが行われています。

学習のためのリソースも増えており、Coursera や Udacity など、オンラインで受講できる講座でも非常に良質な機械学習のチュートリアルに触れることができるようになっています。

- 機械学習を利用したサービスプラットフォームの登場
- オープンソースの機械学習ライブラリの登場により、研究と応用との間が短くなっている
- 学習のためのリソースの拡充

これらを背景とし、「機械学習の適用」はこれまでより非常に行いやすくなっています。 逆に言えば、これからはこれまでとは比較にならないスピードでビジネスへの適用が進んでくると思います。

アートシーン

機械学習は、なにもビジネスだけに使うものではありません。 アート、というとシステムとは縁遠い領域というイメージがあると思いますが、機械学習はアートの領域でも大きな成果を上げています。

機械学習はデータから学習をして推論を行う技術であり、これはアート分野でも例外ではありません。 事前学習の箇所で「画像の特徴を抽出する」ことが可能になったと説明しましたが、これはつまり画像の特徴がモデルの中に表現されているということです。

この学習により獲得した「特徴」とはつまり、画風、メロディ、文体といったものになります。

これを利用することで、写真に画風を適用したり、ありそうなメロディや文書を生成したりすることが可能になってきているということです。

ビジネス上のデータとは異なり、アートに関するデータ(画像や音楽、文書といったもの)は公開されているものが多く、広く収集することが可能です。 機械学習を行うには多くのデータがあるほ

うが有利なため、今後はむしろビジネスよりもアートの方面から革新的な成果が出てくると思います。

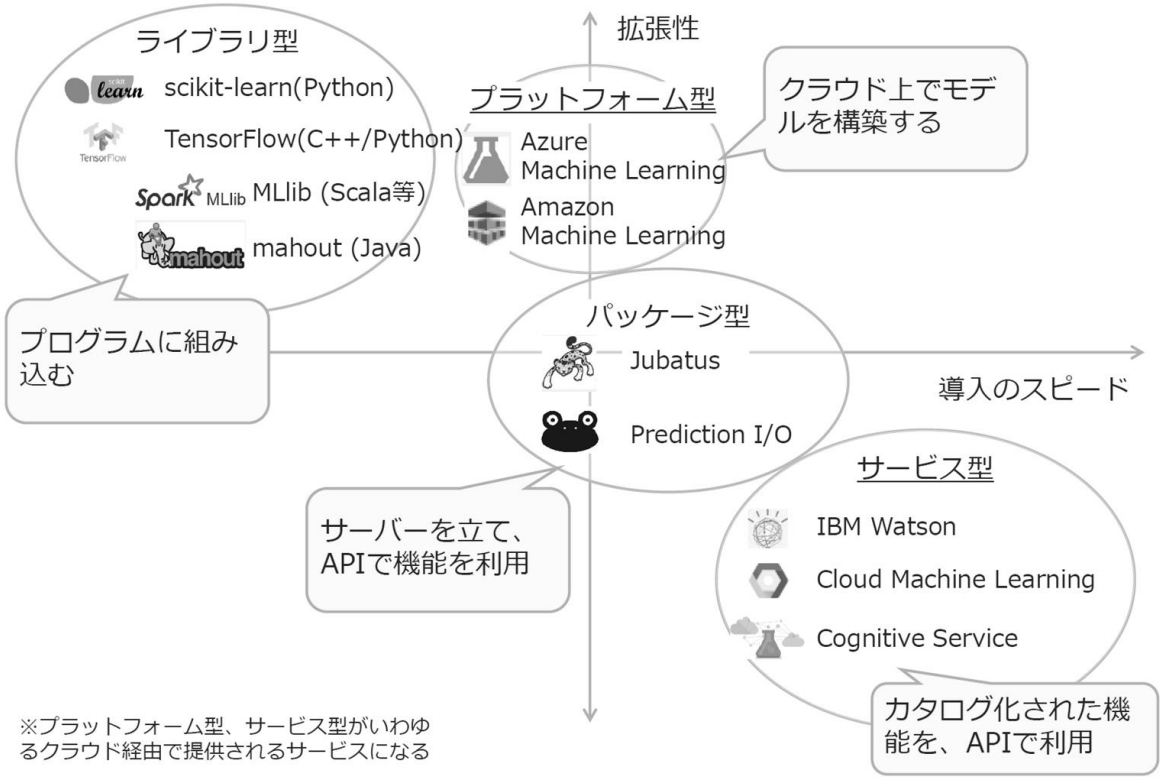
この観点からすれば、業種にかかわらず「大量のデータ」を保有している企業が機械学習によりそれを新しいビジネスに転換するということは十分にあり得ます。　Google　の自動運転車などは、その好例といえると思います。

機械学習の活用を検討することが重要なのはもちろんですが、それにより自社(アーティストであれば自己)の 市場が脅かされる可能性についても、検討しなければならない時代になっているといえると思います。

機械学習を実装する際の選択肢

ここでは、実際に機械学習を利用する際の選択肢について提示します。
実利用例の章でも触れた通り機械学習サービスなどの登場により、機械学習を活用するにはその筋の専門家がいないといけない、 という状況でなはなくなりつつあります。

現時点で選択可能な実装手段を、図として示したものが以下になります。



ライブラリ型、プラットフォーム型、パッケージ型、サービス型の 4 つを提示しています。
これらを、拡張性と導入のスピードの二軸で分類しています。拡張性と導入スピードは基本的にトレードオフの関係です。

- 自由度が高い分、実装には時間・知識が要求され導入スピードが下がる
- 導入スピードが速い分、定型的なものを使わざるを得ず拡張性は低くなる

ここからは、各実装方法の特徴についてこの観点から解説を行っていきます。

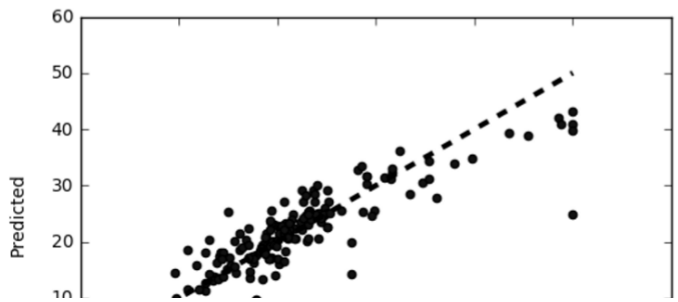
ライブラリ型

ライブラリ型は、プログラムの中から呼び出して機械学習アルゴリズムを実装・利用するタイプです。
様々なプログラミング言語で機械学習が利用できるライブラリが開発されていますが、 最も充実しているのは Python という言語になります。そのため、ライブラリ型を利用する場合まず プログラミング言語としては Python が第一候補になると思います。

○：開発するプログラムの中に組み込めるため、自由度が高い開発が可能
×：扱うに当たっては機械学習に関する一定以上の理解が必要

以下は、Python で scikit-learn というライブラリを呼び出して使っている様子になります。

In [2]:
```
from sklearn import datasets
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
import matplotlib.pyplot as plt

def make_model(X, y):
    lr = linear_model.LinearRegression()
    lr.fit(features, price)
    return lr

def plot(predicted, teacher):
    fig, ax = plt.subplots()
    _range = [teacher.min(), teacher.max()]
    ax.plot(_range, _range, "k--", lw=3)
    ax.scatter(teacher, predicted)
    ax.set_xlabel("Teacher")
    ax.set_ylabel("Predicted")
    plt.show()

boston_house_prices = datasets.load_boston()
features = boston_house_prices.data
price = boston_house_prices.target
train_f, test_f, train_p, test_p = train_test_split(features, price, test_size=0.33, random_state=42)

model = make_model(train_f, train_p)
predicted = model.predict(test_f)

plot(predicted, test_p)
```



プラットフォーム型

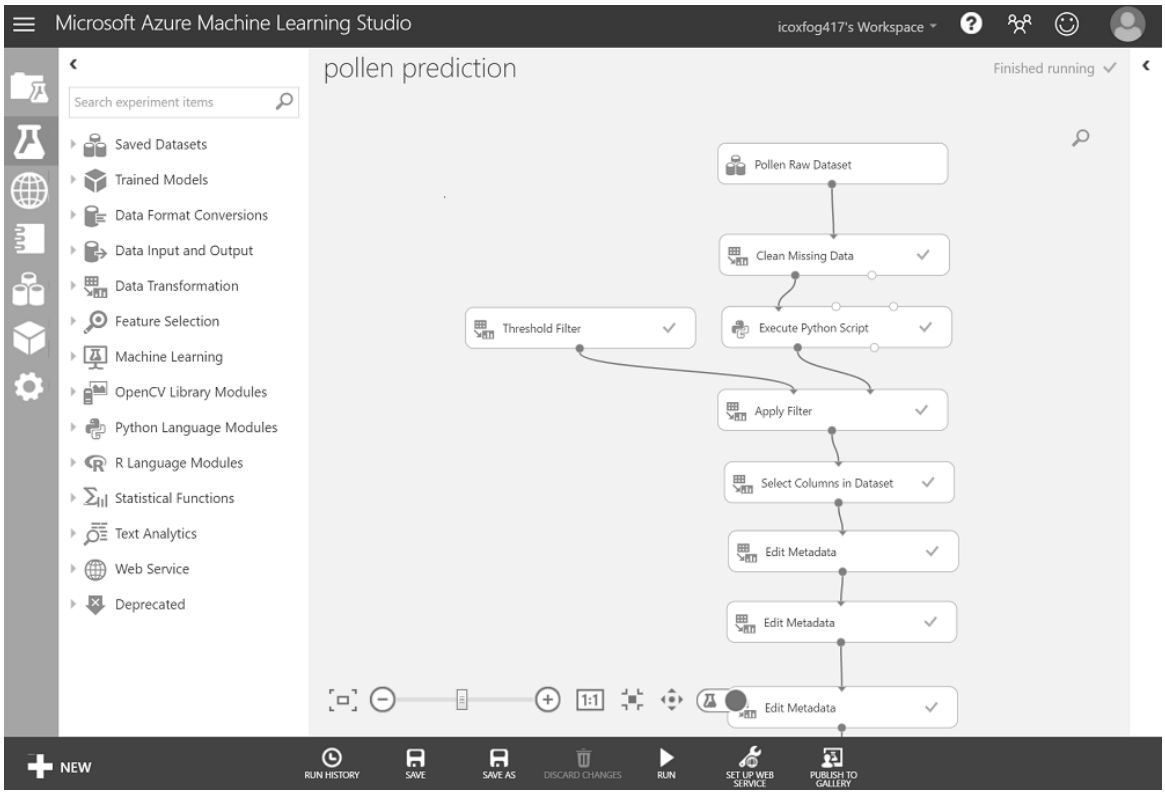
プラットフォーム型は、機械学習モデルの構築をクラウド上で行うタイプです。

モデルの構築という意味ではライブラリ型と同じですが、こちらはクラウド上、つまりブラウザさえあれば作成できるという点が大きなアドバンテージになります。　逆にライブラリ型の場合は、そのライブラリを使うために必要なあれこれをインストールして環境構築を行う必要があります。

ただ、ライブラリ型が自由にモデルの構築を行える半面、プラットフォーム型はプラットフォームで使えるものしか扱うことができません。 これが、トレードオフになります。

○：手元に開発環境を構築する必要がなく、構築したモデルは WebAPI で利用できる
×：開発環境のカスタマイズができず、また用意された処理しか行えない

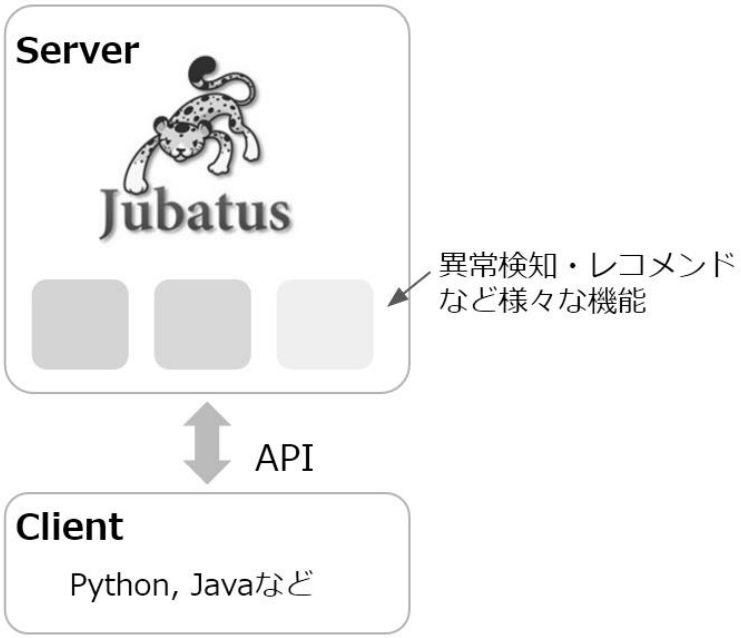
以下は、Azure Machine Learning で実際に作ってみたモデルになります。 左側のメニューからドラッグ&ドロップで作っていくというスタイルになります。



パッケージ型

パッケージ型は、機械学習の活用シーンに適したテンプレートを、API 経由で使用するタイプです。
パッケージ型の特徴は、なんといっても事前構築済みの機能(テンプレート)になります。 これを利用することで、機械学習についての深い知見がなくても、推薦や異常検知といった応用しやすいアルゴリズムを簡単に、手早く使うことができます。

以下は、パッケージ型の一種である Jubatus の例です。



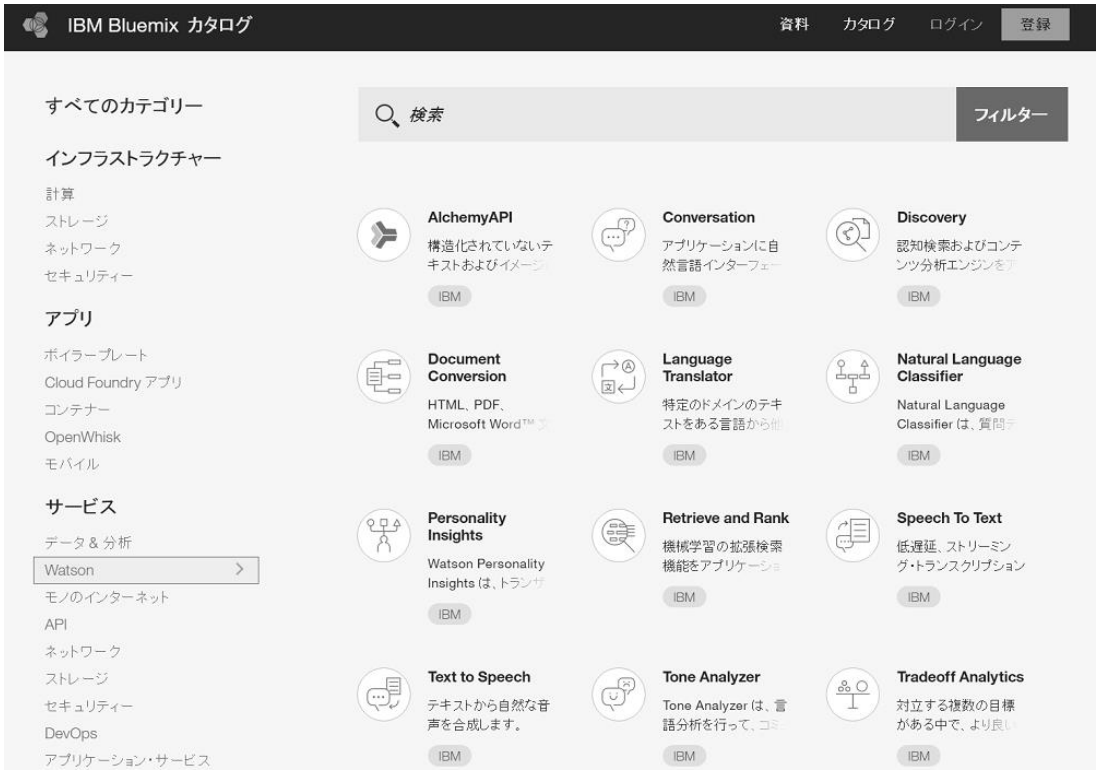
Jubatus は株式会社 Preferred Networks と NTT ソフトウェアイノベーションセンタが共同開発した日本発のオープンソースで、日本語情報・事例も豊富にあります。 パッケージ型を試す場合には、とてもよい入り口であると思います。

いいことづくしに見えますが、パッケージ型は導入のためにサーバーなどのインフラ環境が必要になる点に注意が必要です。

○：機械学習のベストプラクティス集ともいえるテンプレートを、簡単に試すことができる
×：サーバーを構築するための、インフラ環境を用意する必要がある

サービス型

サービス型は、クラウド上で提供されるカタログ化された機能を、WebAPI 経由で使用するタイプです。こちらは見たほうが早いと思います。以下は、IBM Bluemix プラットフォーム上で提供される Watson の API 一覧です。



画像認識やテキスト解析、音声認識といった API が並んでいるかと思います。 サービス型は、このようにカタログ化された機能の中から必要なものを選んで使うタイプになります。
こうしたサービスは紹介した Watson 以外にも Google の Cloud Machine Learning、Microsoft の Cognitive Service など様々なものがあり、 今後も拡充していくと思われます。

サービス型の最大の利点は、すでに高い精度が約束された機能を利用できるという点です。　反面、カタログ式のため機能に融通は聞きません。

○: 品質(=精度)の保証された機能を、簡単に利用することができる
×: 提供されている機能は固定的であり、実際のビジネスには適合させにくい

ニーズに適合するのであれば、高い精度のモデルがすぐに使えるのは大きなメリットです。

実装方式の選択について

結局どれを選べばいいのか、については最初に述べた通り、拡張性と導入スピードのトレードオフを吟味して選択する必要があります。
ただ、ビジネスで使う場合であれば個人的にはパッケージ型をお勧めします。パッケージ型以外を選ばなかった理由については、以下になります。

ライブラリ型

少なくとも、開発担当者に機械学習の知識がなければ、使用するべきではありません。社内に人材がいない場合、採用・育成も込みで考える必要があります

プラットフォーム型

環境が手元の PC かクラウドか、というだけで、求められる知識はライブラリ型とほぼ同等になります

サービス型

手軽に始められるのは良いのですが、機能が固定的で実際のビジネスにはフィットさせにくいです

機械学習の採用にあたっては、まず効果が出るのかを検証するフェーズがあるはずです。 このフェーズで、勉強しながらライブラリ型(プラットフォーム型)を使うのも、業務に合わせにくいサービス型を使うのも多くの場合適切ではありません。

まず典型的な使用例が試せるパッケージ型を使って検証するのが効果的で、その後作りこみが必要と判断されればライブラリ型へ、 サービス型がはまる領域があれば置き換えで、というのが良いのではないかと思います。