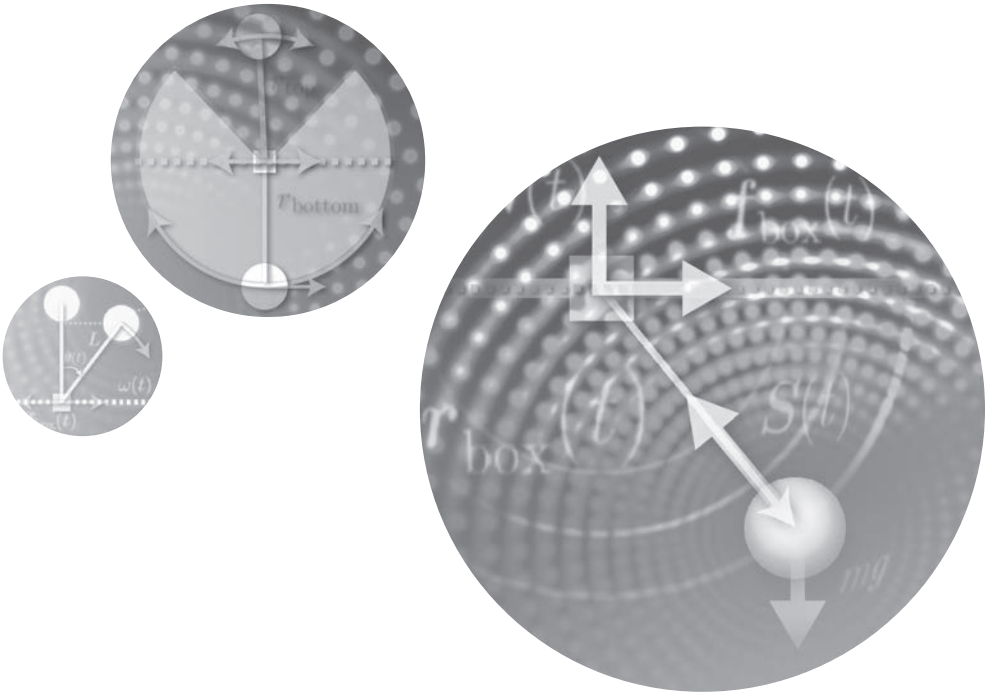
倒立振子の作り方
ゼロから学ぶ強化学習

物理シミュレーション × 機械学習

遠藤理平◉著

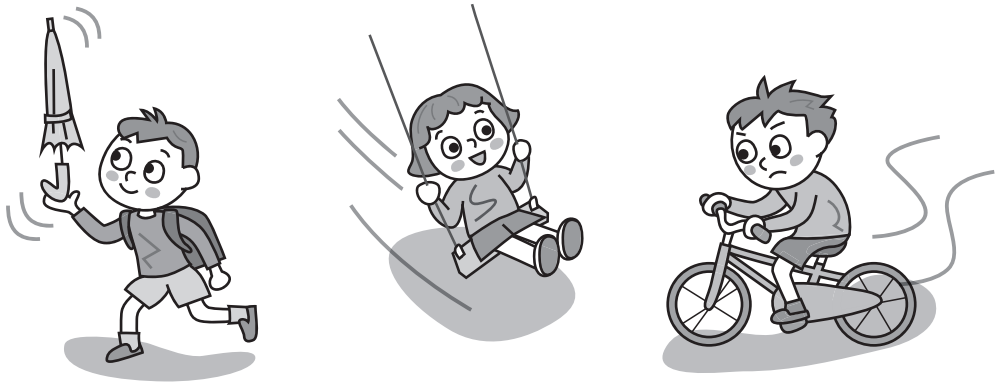


CUTT
カットシステム

はじめに

小学生の時分、雨上がりの学校からの帰り道で、「手のひらに乗せた傘」を倒さないように歩いた経験は誰にでもあると思います。「傘の角度」や手に感じる「傘から受ける力」などの情報をもとに、状況を瞬間的に判断して最適な行動を取ることで傘の状態を維持しますが、誰に教わるでもなく、練習を重ねることで誰でも出来るようになります。反対に、行動指針を言葉で説明しようとすると、冗長でわかりにくい表現にならざるを得ません。

このように言葉で説明するのは難しいが、練習により失敗と成功を繰り返すことで習得できる認知のことを**暗黙知**と呼びます。自転車に乗る、ブランコを漕ぐ、などの動作が、その代表例となります。反対に、言葉や図表、法則などで表現できる知識は**形式知**と呼ばれ、科学・技術と高い親和性がある領域となります。



昨今の人工知能と評される**機械学習**は、人間が設定した目的に対して「試行錯誤の反復訓練」を行うことで、期待値が高い行動を学習できるものです。つまり、コンピュータには苦手な領域とされていた暗黙知を従来の技術基盤に取り込むという、全く新しい価値創造の可能性を意味しています。

本書は、コンピュータを用いて物理現象を再現する「物理シミュレーション」と、与えられた環境内で目的に応じて最適な行動を決定する「強化学習」を組み合わせて解説する書籍です。

先ほど例として紹介した「手のひらに乗せた傘」をモデル化した倒立振子を対象に強化学習の方法を解説していきます。

本書は大きく分けて、前半4章と後半6章の2部構成になっています。前半は、3×3のマス目に○（先手）と×（後手）のマークを交互に埋めていき、「縦・横・斜めのいずれかで同じマークが3つ並ぶと勝ち」という2人対決ゲーム（三目並べ）を題材にして強化学習の基本を解説します。その結果を踏まえて、コンピュータ対戦型の三目並べ（Webブラウザゲーム）を開発します。実行環境にWebブラウザを利用するため、HTML5（JavaScript）を使ってゲームを開発します。

後半は、振り子運動のシミュレーションの実装方法を解説し、その後、倒立振子を強化学習と組み合わせて実現するために必要な要素を順番に解説していきます。物理シミュレーションは計算量が多いため、プログラミング言語としてC++を利用します。

最後に、本書の執筆の機会を頂きました株式会社カットシステムの石塚勝敏さん、非常に丁寧な編集を行なって頂きました阿久澤裕樹さん、また、日常的に議論に付き合って頂いている特定非営利活動法人natural scienceの皆さんに深く感謝申し上げます。

2019年1月　遠藤 理平

◆サンプルファイルのダウンロードについて

本書で解説したサンプルプログラムは、以下のURLからダウンロードできます。強化学習を学ぶときの参考としてご利用ください。

http://----------

動作環境

◆第1章から第4章まで ―――――― HTML5（JavaScript）

HTML5（JavaScript）を用いて、Webブラウザで動作するコンピュータ対戦型の三目並べゲームを開発します。HTML5を用いる利点は大きく分けて3つあります。

1つ目はWebブラウザが動作する環境であれば、Windows／Mac／Androidに限らず、どの端末でも動作するクロスプラットフォーム性です。2つ目は、開発に必要な環境を限定しないことです。HTMLソースを編集する「テキストエディタ」とプログラムを実行する「Webブラウザ」さえあれば、あらためて開発環境を用意する必要はありません。3つ目は、HTMLはもともとWebページ制作用の言語であるため、ユーザーインターフェースの作成が非常に簡単なことです。

本書ではHTML5の記述について詳しく解説していませんが、すべてのソースコードを掲載していますので、わからない箇所はWebなどで確認してみてください。

◆第5章から第10章まで ―――――― C++

本書で紹介するC++のサンプルプログラムは、Visual Studioでコンパイル＆実行することを想定し、Visual Studioソルーションファイルを用意しています。Visual Studioはマイクロソフトが提供する統合開発環境で、様々な言語でアプリケーションを開発することができます。個人利用であれば無償で使用できます（Visual Studio 2017 Community版の場合）。

なお、Visual Studioのほかに、Windowsにおけるgcc実行環境であるMinGW（ver.4.5.0）を用いて、C++のコンパイルならびに実行を確認しています。

・MinGWのgccを使ったコンパイル時の注意点

MinGWで提供されているgcc（ver.5.3.0）のC++コンパイラーは、バージョンがC+98と少し古いタイプになります。このため、C++の最新機能を利用する場合は注意が必要となります。

たとえば、gccでは`std::to_string`を利用できないため（gccの既知のバグ）、整数型から`string`型への型変換を簡単に実行することはできません。C++標準ライブラリの`std::ostringstream`（文字列ストリーム）を利用する必要があります。また、計算結果を出力するファイル名を`ostringstream`クラスの文字列で指定する際に、`ostringstream`クラスの`str`メソッドを利用する必要があります。この`str`メソッドはC+11以降でしか利用できないため、gccコンパイラーでC++をコンパイルする際に、C+11を利用するコンパイルオプションを指定しなければなりません。

なお、gcc（ver.5.3.0）はC++14まで対応しているため(※1)、C++14を利用することにします。コンパイル方法は次のとおりです。

```
g++  ファイル名.cpp  -std=c++14
```

（※1）gccのバージョンとC++コンパイラーのバージョンの対応は公式ページで確認できます。
https://gcc.gnu.org/projects/cxx-status.html

そのほか、gccの場合は、

- 絶対値を計算するabs関数がdouble型に対応していないため、fabs関数を利用する
- 円周率を表す定数M_PIを利用するには、プログラムのはじめに「#define _USE_MATH_DEFINES」を追加する

といったことにも注意しなければなりません。なお、コンパイラーにVisual Studioを利用する場合は、特に注意する必要はありません。

目次

1

強化学習で三目並べを学習させよう！

1.1 強化学習の概念

強化学習とは、ある**環境**（Environment）内にある**エージェント**（Agent）が、現在の**状態**（State）に対して取るべき**行動**（Action）を決定する機械学習の一種です。エージェントが行動すると、それに応じて状態が変化します。同時に、エージェントは行動に対する**報酬**（Reward）を環境から受け取ります。強化学習は、一連の行動によって報酬を最大化する**方策**（Policy）を学習することを指します。三目並べを例にすると、それぞれの対応は以下のようになります。

環境＝「ゲーム全体」
エージェント＝「プレイヤー」
状態＝「譜面」（○×の配置）
行動＝「次の手」
報酬＝「勝敗」
方策＝「勝率を上げるための戦略」

強化学習は、初めから正解がわかっているわけではなく、以下に示した**図1-1**の①～④を何度も繰り返すことで方策を学習していきます。このため、もともと正解が与えられている**「教師あり学習」**とは異なる、**「教師なし学習」**に分類されます。つまり、初めから知られている「良い手」を学習させるのではなく、勝負を繰り返して「勝率の高い手」を経験的に学習させていきます。

正しい方策を導き出すための肝となるのが、目的に応じた報酬の与え方です。報酬の与え方が悪いと目的を達成することができません。プログラマーがすべきことは「良い手」を教えることではなく、報酬の与え方をプログラミングすることになります。

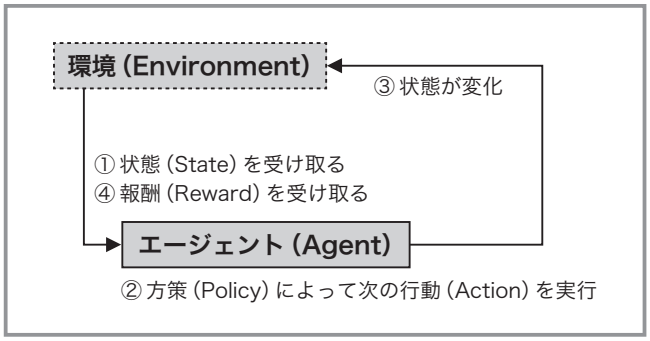


図1-1　強化学習の概念図

本書では、強化学習の中で最も利用されているアルゴリズムの一つである**Q学習**（Q-Learning）を用います。Q学習は、ある状態で取りうる行動のうち「状態に対する行動の価値」が一番高い行動を実行することを方策として学習を行います。この「状態に対する行動の価値」は**行動評価関数**（Q値）と呼ばれ、多次元の表や多変数関数で表現されます。

なお、エージェントは「どのような経緯で現時点の状態に至ったのか」は考慮せず、あくまで「環境から与えられる現時点での状態」に対して、方策に従って次の行動を実行するとします（過去の行動は状態に全て反映されていると考えます）。

1.2 環境・エージェント・状態・行動の定義

三目並べを具体例とした環境、エージェント、状態、行動は以下のとおりです。

■表1-1　環境・エージェント・状態・行動の定義

項目	説明
環境	3×3のマス目の管理と勝敗を判定。ゲーム開始からの手数をtと表す。
エージェント	プレイヤー（先手と後手の2つ）。
状態	全マス目の「○」「×」「未配置」の配置パターン。手数tの状態を$s(t)$と表す。
行動	全マス目の「未配置」のマス目に手を打つ。手数tの行動を$a(s,t)$と表す。

状態と行動について補足説明します。三目並べは0手目から9手目までを個別のパターンとみなすと、3×3＝9つのマス目に「○」「×」「未配置」の3つのどれかが入ると考えられます。マス目の位置まで区別すると全部で$3^9 = 19683$パターンの状態が存在することになります。ただし、三目並べは勝負がついた時点で終了となり、最短で5手目での終了もあり得るため、実際のゲームではこのパターン数よりも少なくなります。

さらには、位置は異なっていても手としては実質的に同じパターンも多数存在します。1手目の状態を示した**図1-2**を例に説明します。先手（○）は全マス目の9箇所どこにでも指すことができますが、本質的に異なる手は「四隅」「四辺の真ん中」「真ん中」の3パターンです。た

とえば、どの四隅に打ったとしても（または、どの四辺の真ん中に打ったとしても）、手としては実質的に同じ状態であると考えられます。このような「実質的に同じ状態」を1つの状態にまとめることで、学習効率を上げることができます。

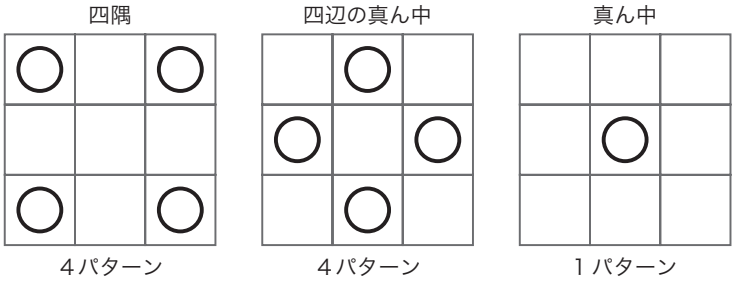


図1-2　本質的に同じ手となるパターン（1手目）

このような「実質的に同じ状態パターン」は、正方形に存在する3つの対称性（線対称・回転対称・点対称）と関係があります。正方形の対称性については2.1節を参照ください。

1.3 状態と行動の三目並べにおける具体例

状態と行動のイメージを沸かせるために、三目並べにおける0手目、1手目、2手目の状態を**図1-3**に示します。$t=0$の状態$s(0)$はまだ何も手が指されていない状態（1パターン）です。エージェントはこの状態を受け取った後に、先手「○」の3つの選択肢から1つの行動$a(0)$を選択して実行します。その結果、状態は$s(0) \rightarrow s(1)$へ遷移します。$s(1)$は前節で解説したように、対称性を考慮すると3パターンです。この各状態に対して、それぞれ後手「×」の選択肢は**図1-3**に示したとおり全部で12パターンあり、そのうち1つの行動$a(1)$を選択して実行します。

実行した結果、状態は$s(1) \rightarrow s(2)$へ遷移します。$s(2)$は対称性を考慮すると12パターンです。次の先手「○」の選択肢は全部で38パターンあり、その中から1つの行動$a(2)$を選択して実行します。このような手順で状態遷移と行動を続けます。

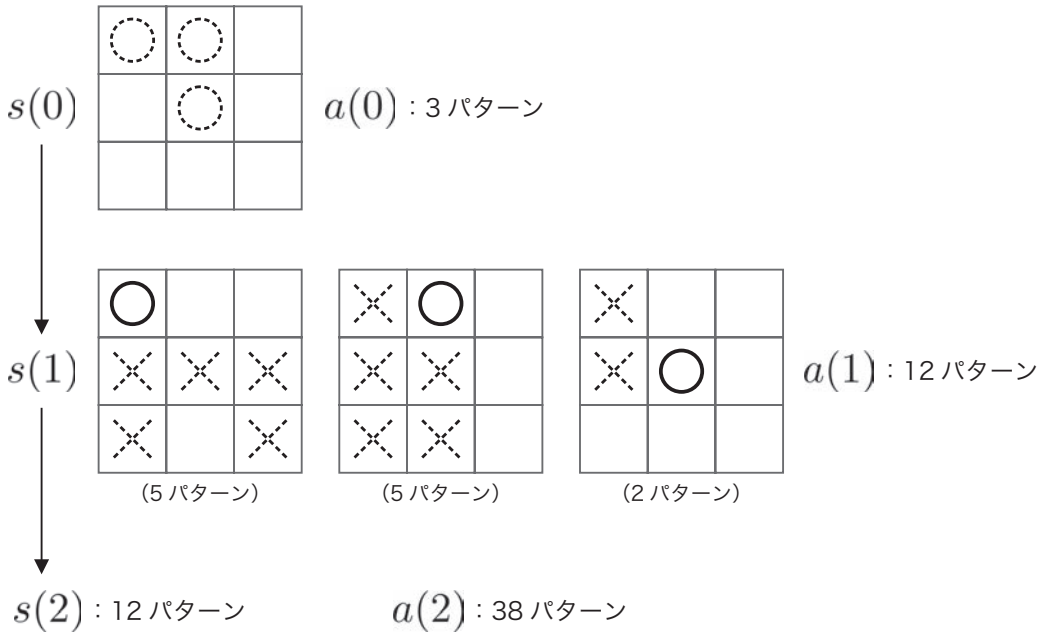


図1-3　状態と行動の具体例（0手目、1手目、2手目）

1.4 報酬の定義

三目並べの勝負の結果は、**図1-4**に示したような先手「○」の勝ち、後手「×」の勝ち、勝負なしのほか、先手「○」が2ライン並んだ勝ちもあり得ます。三目並べのように勝敗がはっきり決定できる場合は、「結果に対して報酬を与える」が最も簡単な報酬の定義となります。勝敗が決定する前の途中の状態は報酬0として、報酬の与え方を**表1-2**のように定義します。なお、t手目の行動$a(t)$で**エージェント**が得られる報酬を$r(t)$と表すことにします。

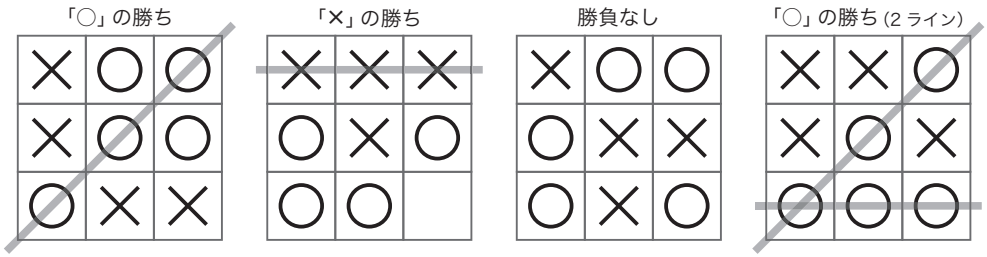


図1-4　三目並べの勝敗パターンの例

■表1-2　三目並べの報酬の例

状態	報酬
途中	0点
勝ち	1点
負け	−1点
勝負なし	0点

ただし、結果が確定した時刻tの報酬$r(t)$のみに値を与えた場合、勝負が決まる前の各手の有利・不利が反映されません。そこで、勝負が決定するまでの途中の報酬も別途、与える必要があります。三目並べの報酬の与え方については3.1節で詳しく議論します。